VS8053b - AUDIO PROCESSOR WITH Ogg Vorbis / FLAC / WAV SUPPORT

Features

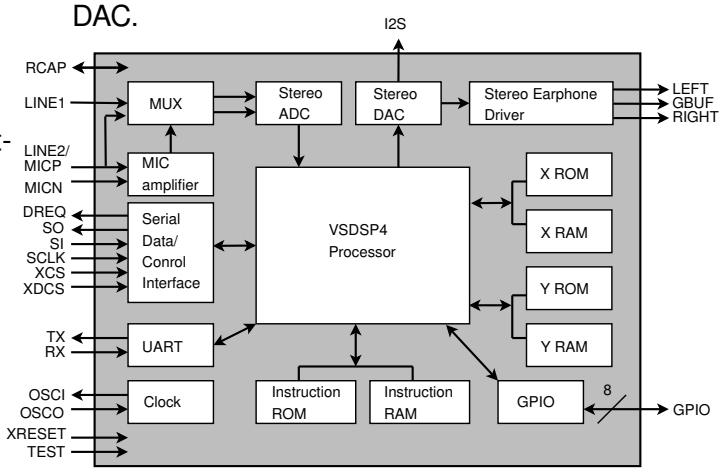
- Laser-fused version of VS1053b
- Decodes
  Ogg Vorbis;
  FLAC with software plugin;
  WAV (PCM + IMA ADPCM)
- Encodes Ogg Vorbis w/ software plugin
- Encodes stereo IMA ADPCM / PCM
- Streaming support for WAV
- EarSpeaker Spatial Processing
- Bass and treble controls
- Operates with a single 12..13 MHz clock
- Can also be used with a 24..26 MHz clock
- Internal PLL clock multiplier
- Low-power operation
- High-quality on-chip stereo DAC with no phase error between channels
- Zero-cross detection for smooth volume change
- Stereo earphone driver capable of driving a 30 Ω load
- Quiet power-on and power-off
- I2S output interface for external DAC
- Separate voltages for analog, digital, I/O
- On-chip RAM for user code and data
- Serial control and data interfaces
- Can be used as a slave co-processor
- SPI flash boot for special applications
- UART for debugging purposes
- New functions may be added with software and up to 8 GPIO pins
- Lead-free RoHS-compliant package

Description

VS8053b is an audio processor with a high performance stereo ADC and DAC. VS8053b supports audio formats that do not require a license (Ogg Vorbis, PCM, IMA-ADPCM, FLAC). By using the Ogg Vorbis encoder plugin the device can be used to build a high performance and low cost digital audio encoder-decoder link.

VS8053b contains a high-performance, proprietary low-power DSP processor core VS_DSP^4, 16 KiB instruction RAM and 16 KiB data RAM for user applications running simultaneously with built-in decoders, serial control and input data interfaces, up to 8 general purpose I/O pins, an UART, as well as a high-quality variable-sample-rate stereo ADC (mic, line, line + mic or 2×line) and stereo DAC, followed by an earphone amplifier and a common voltage buffer.

VS8053b receives its input bitstream through a serial input bus, which it listens to as a system slave. The input stream is processed and passed through a asynchronous sample rate converter and digital volume control to an 18-bit oversampling, multi-bit, sigma-delta DAC.



Contents

List of Figures

1 Licenses

To the best of our knowledge, this device does not contain any coding software that needs a license.

2 Disclaimer

All properties and figures are subject to change.

3 Definitions

ABR Average BitRate. Bitrate of stream may vary locally, but stays close to a given number when averaged over a longer time.

B Byte, 8 bits.

b Bit.

CBR Constant BitRate. Bitrate of stream is the same for each compression block.

CBUF Headphone Common Buffer. Outputs DC voltage.

GBUF Same as CBUF.

Ki "Kibi" = 2^{10} = 1024 (IEC 60027-2).

Mi "Mebi" = 2^{20} = 1048576 (IEC 60027-2).

SCI Serial Control Interface, an SPI bus for VS8053b control.

SDI Serial Data Interface, an SPI bus for VS8053b bitstream data.

VBR Variable BitRate. Bitrate varies depending on the complexity of the source material.

VS_DSP VLSI Solution's DSP core.

VSIDE VLSI Solution's Integrated Development Environment.

W Word. In VS_DSP, instruction words are 32 bits and data words are 16 bits wide.

4 Characteristics & Specifications

4.1 Absolute Maximum Ratings

Parameter	Symbol	Min	Max	Unit
Analog Positive Supply	AVDD	-0.3	3.6	V
Digital Positive Supply	CVDD	-0.3	1.85	V
I/O Positive Supply	IOVDD	-0.3	3.6	V
Current at Any Non-Power Pin[1]			± 50	mA
Voltage at Any Digital Input		-0.3	IOVDD+0.3[2]	V
Operating Temperature		-40	+85	°C
Storage Temperature		-65	+150	°C

[1] Higher current can cause latch-up.
[2] Must not exceed 3.6 V

4.2 Recommended Operating Conditions

Parameter	Symbol	Min	Typ	Max	Unit
Ambient Operating Temperature		-40		+85	°C
Analog and Digital Ground [1]	AGND DGND		0.0		V
Positive Analog, VREF=1.23V [2]	AVDD12	2.6	2.8	3.6	V
Positive Analog, VREF=1.65V [2]	AVDD16	3.3	3.3	3.6	V
Positive Digital	CVDD	1.7	1.8	1.85	V
I/O Voltage	IOVDD	1.8	2.8	3.6	V
Input Clock Frequency [3]	XTALI	12	12.288	13	MHz
Internal Clock Frequency	CLKI	12	36.864	55.3	MHz
Internal Clock Multiplier [4]	CLKM	$1.0\times$	$3.0\times$	$4.5\times$	
Master Clock Duty Cycle		40	50	60	%

[1] Must be connected together as close the device as possible for latch-up immunity.
[2] Reference voltage can be internally selected between 1.23V and 1.65V, see section 9.6.2.
[3] The maximum sample rate that can be played with correct speed is XTALI/256 (or XTALI/512 if SM_CLK_RANGE is set). Thus, XTALI must be at least 12.288 MHz (24.576 MHz) to be able to play 48 kHz at correct speed.
[4] Reset value is $1.0\times$. Recommended SC_MULT=$3.5\times$, SC_ADD=$1.0\times$ (SCI_CLOCKF=0x8800). Do not exceed maximum specification for CLKI.

4.3 Analog Characteristics

Unless otherwise noted: AVDD=3.3V, CVDD=1.8V, IOVDD=2.8V, REF=1.65V, TA=-30..+85°C, XTALI=12..13MHz, Internal Clock Multiplier $3.5\times$. DAC tested with 1307.894 Hz full-scale output sinewave, measurement bandwidth 20..20000 Hz, analog output load: LEFT to GBUF 30 Ω, RIGHT to GBUF 30 Ω. Microphone test amplitude 48 mVpp (differential), f_s=1 kHz, Line input test amplitude 2.52 Vpp, f_s=1 kHz.

DAC Characteristics					
Parameter	**Symbol**	**Min**	**Typ**	**Max**	**Unit**
DAC Resolution			18		bits
Total Harmonic Distortion, -3 dB of full-scale	THD			0.04	%
Third Harmonic Distortion, -3 dB of full-scale				0.01	%
Dynamic Range (DAC unmuted, A-weighted)	IDR		100		dB
S/N Ratio (full scale signal)	SNR		94		dB
Interchannel Isolation (Cross Talk), 600Ω + GBUF			80		dB
Interchannel Isolation (Cross Talk), 30Ω + GBUF			53		dB
Interchannel Gain Mismatch		-0.5		0.5	dB
Frequency Response		-0.1		0.1	dB
Full Scale Output Voltage	LEVEL16		2750[1]		mVpp
Full Scale Output Voltage, VREF = 1.2 V	LEVEL12		2050[1]		mVpp
Deviation from Linear Phase				5	°
Analog Output Load Resistance	AOLR	16	30[2]		Ω
Analog Output Load Capacitance				100	pF
DC level (CBUF, LEFT, RIGHT)	VREF16		1.65		V
DC level (CBUF, LEFT, RIGHT), VREF = 1.2 V	VREF12		1.23		V

[1] double can be achieved with +-to-+ wiring for mono difference sound.
[2] AOLR may be much lower, but below *Typical* distortion performance may be compromised.

ADC Characteristics					
Parameter	**Symbol**	**Min**	**Typ**	**Max**	**Unit**
Microphone input amplifier gain	MGAIN		26		dB
Microphone input amplitude (differential)	MLEV16		64	180[1]	mVpp AC
Microphone input amplitude (diff.), VREF = 1.2 V	MLEV12		48	140[1]	mVpp AC
Microphone Total Harmonic Distortion	MTHD		0.03	0.07	%
Microphone S/N Ratio	MSNR	60	72		dB
Microphone input impedances, per pin	MIMP		45		kΩ
Line input amplitude	LLEV16		2500	2800[1]	mVpp AC
Line input amplitude, VREF = 1.2 V	LLEV12		1900	2100[1]	mVpp AC
Line input Total Harmonic Distortion	LTHD		0.005	0.014	%
Line input S/N Ratio	LSNR	85	90		dB
Line input impedance	LIMP		80		kΩ

[1] Harmonic Distortion increases above typical amplitude.

4.4 Power Consumption

Internal clock multiplier $3.0\times$. TA=+25°C. IOVDD =2.8 V, AVDD = 2.6 V, CVDD = 1.8V̇.

XRESET active				
Parameter	**Min**	**Typ**	**Max**	**Unit**
Current IOVDD		0.3	3.0	μA
Current AVDD		0.6	5.0	μA
Current CVDD		18	35.0	μA
Full-scale sine in sine test mode				
Parameter	**Min**	**Typ**	**Max**	**Unit**
Power Supply Consumption AVDD, no load		5		mA
Current AVDD, output load 30 Ω + GBUF	30	37	60	mA
Current CVDD	8	10	15	mA
128 kbit/s Ogg Vorbis audio plaback, full volume				
Parameter	**Min**	**Typ**	**Max**	**Unit**
Power Supply Consumption AVDD, no load		5		mA
Current AVDD, output load 30 Ω		11		mA
Current AVDD, output load 30 Ω + GBUF		11		mA
Current CVDD		11		mA

4.5 Digital Characteristics

Parameter	**Min**	**Max**	**Unit**
High-Level Input Voltage (xRESET, XTALI, XTALO)	$0.7\times$IOVDD	IOVDD+0.3[1]	V
High-Level Input Voltage (other input pins)	$0.7\times$CVDD	IOVDD+0.3[1]	V
Low-Level Input Voltage	-0.2	$0.3\times$CVDD	V
High-Level Output Voltage at XTALO = -0.1 mA	$0.7\times$IOVDD		V
Low-Level Output Voltage at XTALO = 0.1 mA		$0.3\times$IOVDD	V
High-Level Output Voltage at I_O = -1.0 mA	$0.7\times$IOVDD		V
Low-Level Output Voltage at I_O = 1.0 mA		$0.3\times$IOVDD	V
Input Leakage Current	-1.0	1.0	μA
SPI Input Clock Frequency [2]		$\frac{CLKI}{7}$	MHz
Rise time of all output pins, load = 50 pF		50	ns

[1] Must not exceed 3.6V
[2] Value for SCI reads. SCI and SDI writes allow $\frac{CLKI}{4}$.

4.6 Switching Characteristics - Boot Initialization

Parameter	**Symbol**	**Min**	**Max**	**Unit**
XRESET active time		2		XTALI
XRESET inactive to software ready		22000	50000[1]	XTALI
Power on reset, rise time to CVDD		10		V/s

[1] DREQ rises when initialization is complete. You should not send any data or commands before that.

5 Packages and Pin Descriptions

5.1 Packages

LPQFP-48 is a lead (Pb) free and also RoHS compliant package. RoHS is a short name of *Directive 2002/95/EC on the restriction of the use of certain hazardous substances in electrical and electronic equipment*.

5.1.1 LQFP-48

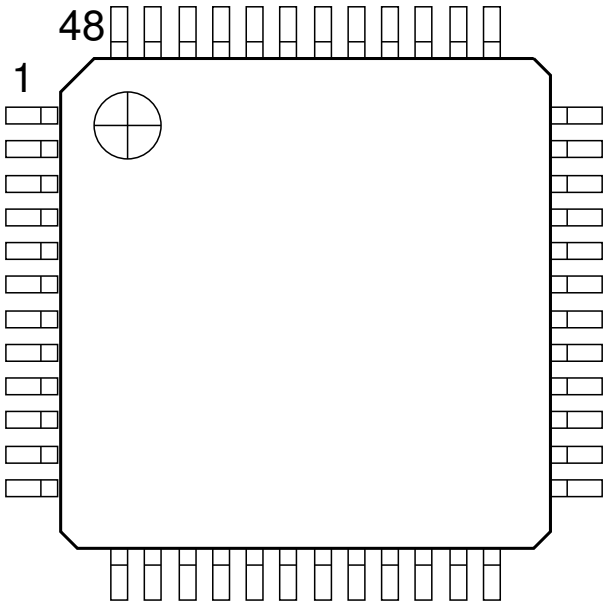


Figure 1: Pin configuration, LQFP-48.

LQFP-48 package dimensions are at *http://www.vlsi.fi/* .

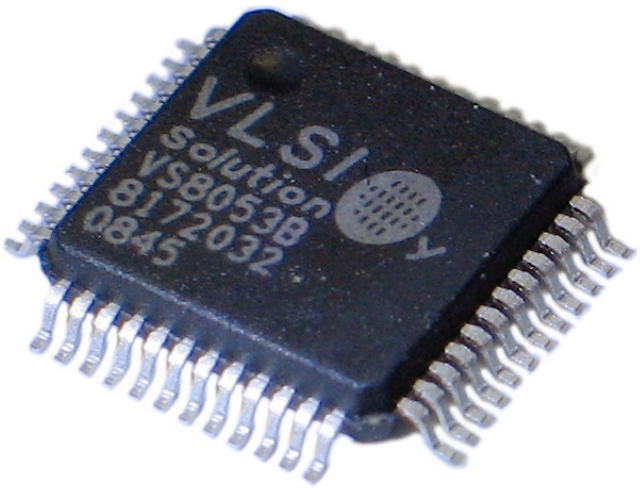


Figure 2: VS8053b in LQFP-48 packaging.

Pad Name	LQFP Pin	Pin Type	Function
MICP / LINE1	1	AI	Positive differential mic input, self-biasing / Line-in 1
MICN	2	AI	Negative differential mic input, self-biasing
XRESET	3	DI	Active low asynchronous reset, schmitt-trigger input
DGND0	4	DGND	Core & I/O ground
CVDD0	5	CPWR	Core power supply
IOVDD0	6	IOPWR	I/O power supply
CVDD1	7	CPWR	Core power supply
DREQ	8	DO	Data request, input bus
GPIO2 / DCLK[1]	9	DIO	General purpose IO 2 / serial input data bus clock
GPIO3 / SDATA[1]	10	DIO	General purpose IO 3 / serial data input
GPIO6 / I2S_SCLK[3]	11	DIO	General purpose IO 6 / I2S_SCLK
GPIO7 / I2S_SDATA[3]	12	DIO	General purpose IO 7 / I2S_SDATA
XDCS / BSYNC[1]	13	DI	Data chip select / byte sync
IOVDD1	14	IOPWR	I/O power supply
VCO	15	DO	For testing only (Clock VCO output)
DGND1	16	DGND	Core & I/O ground
XTALO	17	AO	Crystal output
XTALI	18	AI	Crystal input
IOVDD2	19	IOPWR	I/O power supply
DGND2	20	DGND	Core & I/O ground
DGND3	21	DGND	Core & I/O ground
DGND4	22	DGND	Core & I/O ground
XCS	23	DI	Chip select input (active low)
CVDD2	24	CPWR	Core power supply
GPIO5 / I2S_MCLK[3]	25	DIO	General purpose IO 5 / I2S_MCLK
RX	26	DI	UART receive, connect to IOVDD if not used
TX	27	DO	UART transmit
SCLK	28	DI	Clock for serial bus
SI	29	DI	Serial input
SO	30	DO3	Serial output
CVDD3	31	CPWR	Core power supply
XTEST	32	DI	Reserved for test, connect to IOVDD
GPIO0	33	DIO	Gen. purp. IO 0 (SPIBOOT), use 100 kΩ pull-down resistor[2]
GPIO1	34	DIO	General purpose IO 1
GND	35	DGND	I/O Ground
GPIO4 / I2S_LROUT[3]	36	DIO	General purpose IO 4 / I2S_LROUT
AGND0	37	APWR	Analog ground, low-noise reference
AVDD0	38	APWR	Analog power supply
RIGHT	39	AO	Right channel output
AGND1	40	APWR	Analog ground
AGND2	41	APWR	Analog ground
GBUF	42	AO	Common buffer for headphones, do NOT connect to ground!
AVDD1	43	APWR	Analog power supply
RCAP	44	AIO	Filtering capacitance for reference
AVDD2	45	APWR	Analog power supply
LEFT	46	AO	Left channel output
AGND3	47	APWR	Analog ground
LINE2	48	AI	Line-in 2 (right channel)

[1] First pin function is active in New Mode, latter in Compatibility Mode.

[2] If GPIO0 is high, SPI Boot is tried. See Chapter 10.9 for details.

[3] If GPIO0 is low and GPIO1 is high, behaviour of VS8053 is undefined.

[4] If I2S_CF_ENA is '0' the pins are used for GPIO. See Chapter 11.14 for details.

Pin types:

Type	Description
DI	Digital input, CMOS Input Pad
DO	Digital output, CMOS Input Pad
DIO	Digital input/output
DO3	Digital output, CMOS Tri-stated Output Pad
AI	Analog input

Type	Description
AO	Analog output
AIO	Analog input/output
APWR	Analog power supply pin
DGND	Core or I/O ground pin
CPWR	Core power supply pin
IOPWR	I/O power supply pin

6 Connection Diagram, LQFP-48

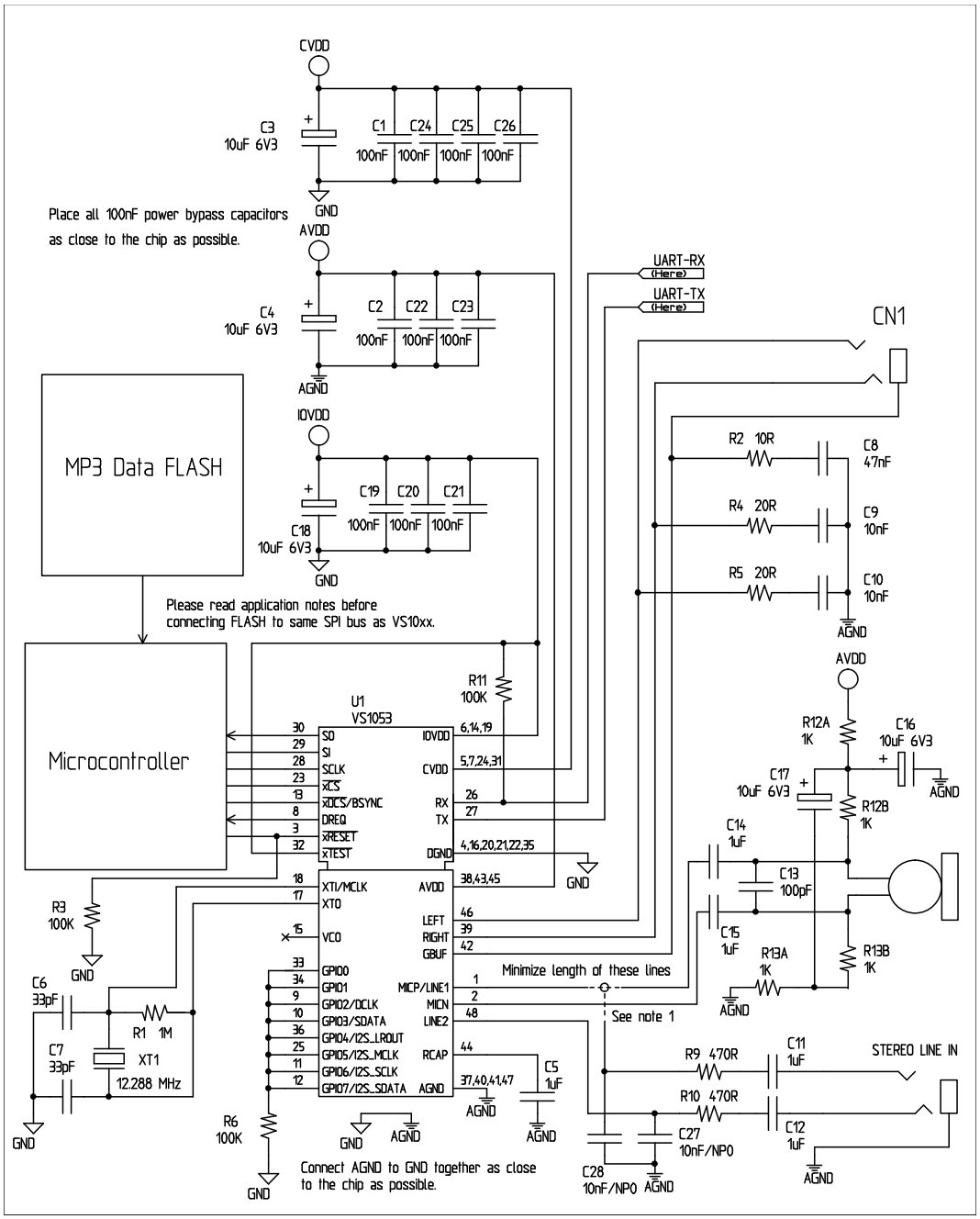


Figure 3: Typical connection diagram using LQFP-48.

Figure 3 shows a typical connection diagram for VS8053.

Figure Note 1: Connect either Microphone In or Line In, but not both at the same time.

Note: This connection assumes SM_SDINEW is active (see Chapter 9.6.1). If also SM_SDISHARE is used, xDCS should be tied low or high (see Chapter 7.1.1).

The common buffer GBUF can be used for common voltage (1.23 V) for earphones. This eliminates the need for large isolation capacitors on line outputs, and thus the audio output pins from VS8053b may be connected directly to the earphone connector.

GBUF must NOT be connected to ground under any circumstances. If GBUF is not used, LEFT and RIGHT must be provided with coupling capacitors. To keep GBUF stable, you should always have the resistor and capacitor even when GBUF is not used. See application notes for details.

Unused GPIO pins should have a pull-down resistor. Unused line and microphone inputs should not be connected.

If UART is not used, RX should be connected to IOVDD and TX be unconnected.

Do not connect any external load to XTALO.

7 SPI Buses

The SPI Bus - which was originally used in some Motorola devices - has been used for both VS8053b's Serial Data Interface SDI (Chapters 7.3 and 9.4) and Serial Control Interface SCI (Chapters 7.4 and 9.5).

7.1 SPI Bus Pin Descriptions

7.1.1 VS10xx Native Modes (New Mode, recommended)

These modes are active on VS8053b when SM_SDINEW is set to 1 (default at startup). DCLK and SDATA are not used for data transfer and they can be used as general-purpose I/O pins (GPIO2 and GPIO3). BSYNC function changes to data interface chip select (XDCS).

SDI Pin	SCI Pin	Description
XDCS	XCS	Active low chip select input. A high level forces the serial interface into standby mode, ending the current operation. A high level also forces serial output (SO) to high impedance state. If SM_SDISHARE is 1, pin XDCS is not used, but the signal is generated internally by inverting XCS.
SCK		Serial clock input. SCK can be gated or continuous. In either case, the first rising clock edge after XCS has gone low marks the first bit to be written.
SI		Serial input. If a chip select is active, SI is sampled on the rising CLK edge.
-	SO	Serial output. In reads, data is shifted out on the falling SCK edge. In writes SO is at a high impedance state.

7.1.2 VS1001 Compatibility Mode (deprecated, do not use in new designs)

This mode is active when SM_SDINEW is set to 0. In this mode, DCLK, SDATA and BSYNC are active.

SDI Pin	SCI Pin	Description
-	XCS	Active low chip select input. A high level forces the serial interface into standby mode, ending the current operation. A high level also forces serial output (SO) to high impedance state.
BSYNC	-	SDI data is synchronized with a rising edge of BSYNC.
DCLK	SCK	Serial clock input. SCK can be gated or continuous. In either case, the first rising clock edge after XCS has gone low marks the first bit to be written.
SDATA	SI	Serial input. SI is sampled on the rising SCK edge, if XCS is low.
-	SO	Serial output. In reads, data is shifted out on the falling SCK edge. In writes SO is at a high impedance state.

7.2 Data Request Pin DREQ

The DREQ pin/signal is used to signal if VS8053b's 2048-byte FIFO is capable of receiving data. If DREQ is high, VS8053b can take at least 32 bytes of SDI data or one SCI command. DREQ is turned low when the stream buffer is too full and for the duration of an SCI command.

Because of the 32-byte safety area, the sender may send up to 32 bytes of SDI data at a time without checking the status of DREQ, making controlling VS8053b easier for low-speed microcontrollers.

Note: DREQ may turn low or high at any time, even during a byte transmission. Thus, DREQ should only be used to decide whether to send more bytes. A transmission that has already started doesn't need to be aborted.

Note: In VS8053b DREQ also goes down while an SCI operation is in progress.

There are cases when you still want to send SCI commands when DREQ is low. Because DREQ is shared between SDI and SCI, you can not determine if an SCI command has been executed if SDI is not ready to receive data. In this case you need a long enough delay after every SCI command to make certain none of them are missed. The SCI Registers table in Chapter 9.6 gives the worst-case handling time for each SCI register write.

Note: The status of DREQ can also be read through SCI with the following code. For details on SCI registers, see Chapter 7.4.

```
// This example reads status of DREQ pin through the SPI/SCI register
// interface.
#define SCI_WRAMADDR 7
#define SCI_WRAM 6
while (!endOfFile) {
  int dreq;
  WriteSciReg(SCI_WRAMADDR, 0xC012); // Send address of DREQ register
  dreq = ReadSciReg(SCI_WRAM) & 1;   // Read value of DREQ (in bit 0)
  if (dreq) {
    // DREQ high: send 1-32 bytes audio data
  } else {
    // DREQ low: wait 5 milliseconds (so that VS10xx doesn't get
    // continuous SCI operations)
  }
} /* while (!endOfFile) */
```

7.3 Serial Protocol for Serial Data Interface (SPI / SDI)

The serial data interface operates in slave mode so DCLK signal must be generated by an external circuit.

Data (SDATA signal) can be clocked in at either the rising or falling edge of DCLK (Chapter 9.6).

VS8053b assumes its data input to be byte-sychronized. SDI bytes may be transmitted either MSb or LSb first, depending of register SCI_MODE bit SM_SDIORD (Chapter 9.6.1).

The firmware is able to accept the maximum bitrate the SDI supports.

7.3.1 SDI in VS10xx Native Modes (New Mode, recommended)

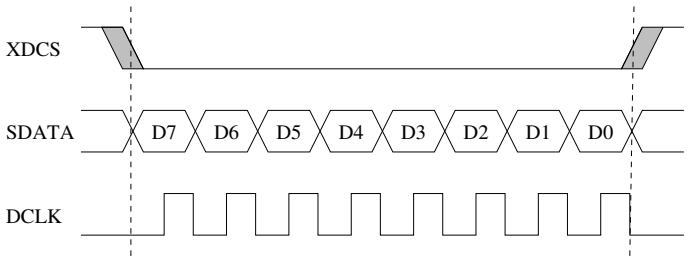


Figure 4: SDI in VS10xx Native Mode, single-byte transfer

In VS10xx native modes (SM_NEWMODE is 1), byte synchronization is achieved by XDCS, as shown in Figure 4. The state of XDCS may not change while a data byte transfer is in progress. XDCS does not need to be deactivated and reactivated for every byte transfer, as shown in Figure 5. However, to maintain data synchronization even if there are occasional clock glitches, it is recommended to deactivate and reactivate XDCS every now and then, for example after each 32 bytes of data.

Note that when sending data through SDI you have to check the Data Request Pin DREQ at least after every 32 bytes (Chapter 7.2).

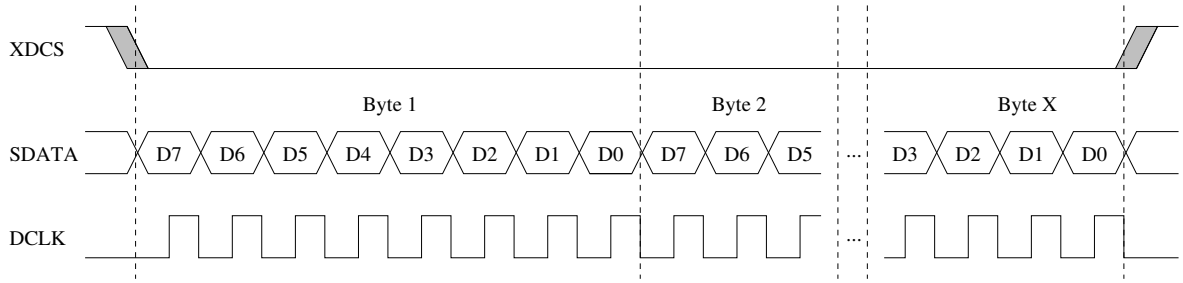


Figure 5: SDI in VS10xx Native Mode, multi-byte transfer, $X \geq 1$

If SM_SDISHARE is 1, the XDCS signal is internally generated by inverting the XCS input.

7.3.2 SDI Timing Diagram in VS10xx Native Modes (New Mode)

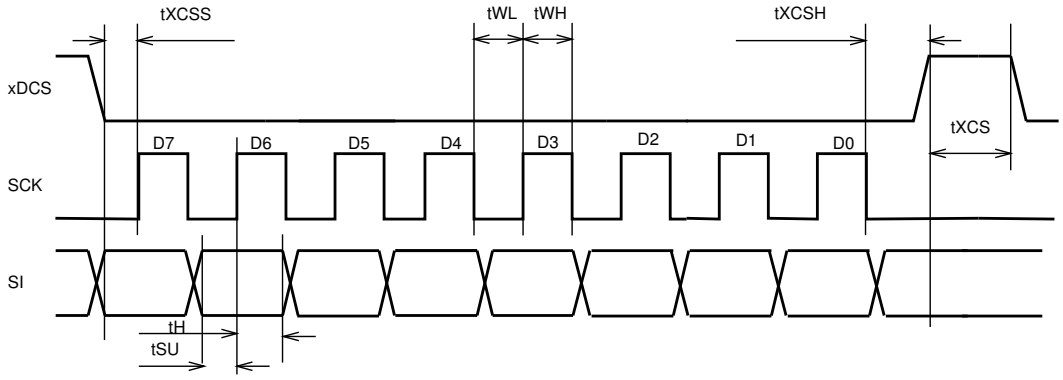


Figure 6: SDI timing diagram

Figure 6 presents SDI bus timing.

Symbol	Min	Max	Unit
tXCSS	5		ns
tSU	0		ns
tH	2		CLKI cycles
tWL	2		CLKI cycles
tWH	2		CLKI cycles
tXCSH	1		CLKI cycles
tXCS	0		CLKI cycles

Note: xDCS is not required to go high between bytes, so tXCS is 0.

Note: Although the timing is derived from the internal clock CLKI, the system always starts up in $1.0\times$ mode, thus CLKI=XTALI. After you have configured a higher clock through SCI_CLOCKF and waited for DREQ to rise, you can use a higher SPI speed as well.

7.3.3 SDI in VS1001 Compatibility Mode (deprecated, do not use in new designs)

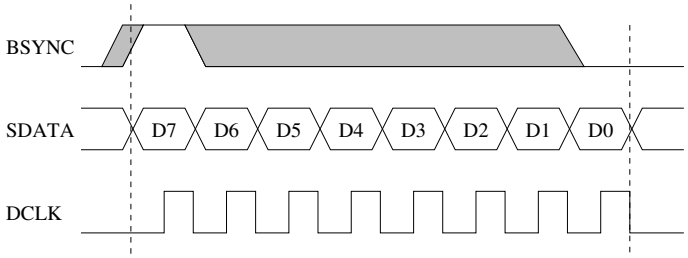


Figure 7: SDI in VS1001 Mode - one byte transfer. Do not use in new designs!

When VS8053b is running in VS1001 compatibility mode, a BSYNC signal must be generated to ensure correct bit-alignment of the input bitstream, as shown in Figures 7 and 8.

The first DCLK sampling edge (rising or falling, depending on selected polarity), during which the BSYNC is high, marks the first bit of a byte (LSB, if LSB-first order is used, MSB, if MSB-first order is used). If BSYNC is '1' when the last bit is received, the receiver stays active and next 8 bits are also received.

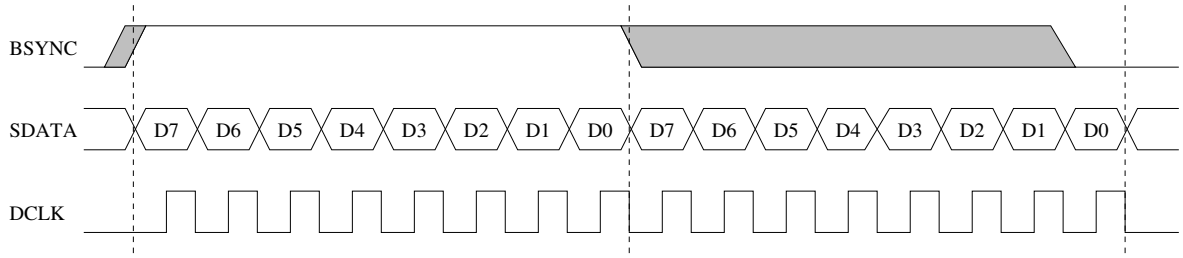


Figure 8: SDI in VS1001 Mode - two byte transfer. Do not use in new designs!

7.3.4 Passive SDI Mode (deprecated, do not use in new designs)

If SM_NEWMODE is 0 and SM_SDISHARE is 1, the operation is otherwise like the VS1001 compatibility mode, but bits are only received while the BSYNC signal is '1'. Rising edge of BSYNC is still used for synchronization.

7.4 Serial Protocol for Serial Command Interface (SPI / SCI)

The serial bus protocol for the Serial Command Interface SCI (Chapter 9.5) consists of an instruction byte, address byte and one 16-bit data word. Each read or write operation can read or write a single register. Data bits are read at the rising edge, so the user should update data at the falling edge. Bytes are always send MSb first. XCS should be low for the full duration of the operation, but you can have pauses between bits if needed.

The operation is specified by an 8-bit instruction opcode. The supported instructions are read and write. See table below.

Instruction		
Name	Opcode	Operation
READ	0b0000 0011	Read data
WRITE	0b0000 0010	Write data

Note: VS8053b sets DREQ low after each SCI operation. The duration depends on the operation. It is not allowed to finish a new SCI/SDI operation before DREQ is high again.

7.4.1 SCI Read

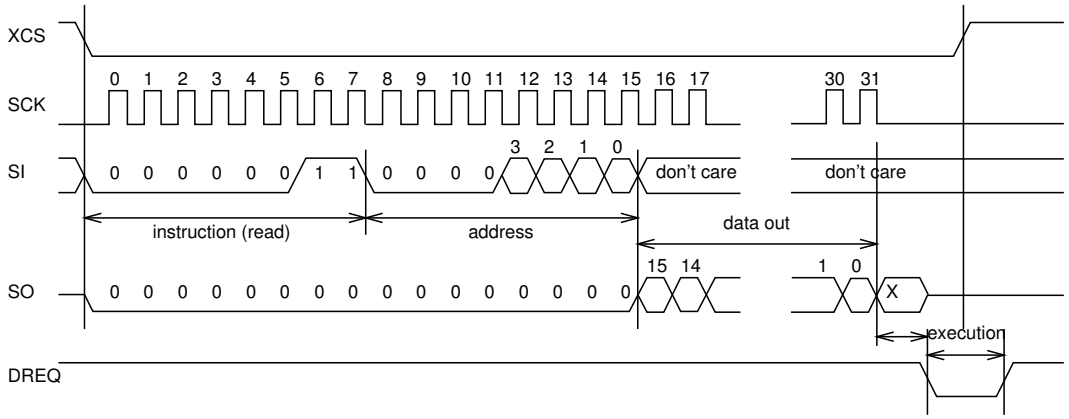


Figure 9: SCI word read

VS8053b registers are read from using the following sequence, as shown in Figure 9. First, XCS line is pulled low to select the device. Then the READ opcode (0x3) is transmitted via the SI line followed by an 8-bit word address. After the address has been read in, any further data on SI is ignored by the chip. The 16-bit data corresponding to the received address is shifted out onto the SO line.

XCS should be driven high after data has been shifted out.

DREQ is driven low for a short while when in a read operation by the chip. This is a very short time and doesn't require special user attention.

7.4.2 SCI Write

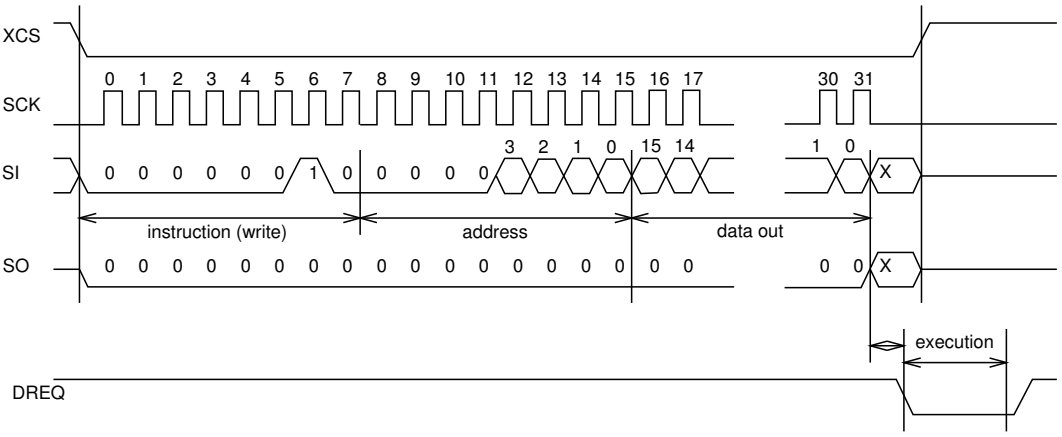


Figure 10: SCI word write

VS8053b registers are written from using the following sequence, as shown in Figure 10. First, XCS line is pulled low to select the device. Then the WRITE opcode (0x2) is transmitted via the SI line followed by an 8-bit word address.

After the word has been shifted in and the last clock has been sent, XCS should be pulled high to end the WRITE sequence.

After the last bit has been sent, DREQ is driven low for the duration of the register update, marked “execution” in the figure. The time varies depending on the register and its contents (see table in Chapter 9.6 for details). If the maximum time is longer than what it takes from the microcontroller to feed the next SCI command or SDI byte, status of DREQ must be checked before finishing the next SCI/SDI operation.

7.4.3 SCI Multiple Write

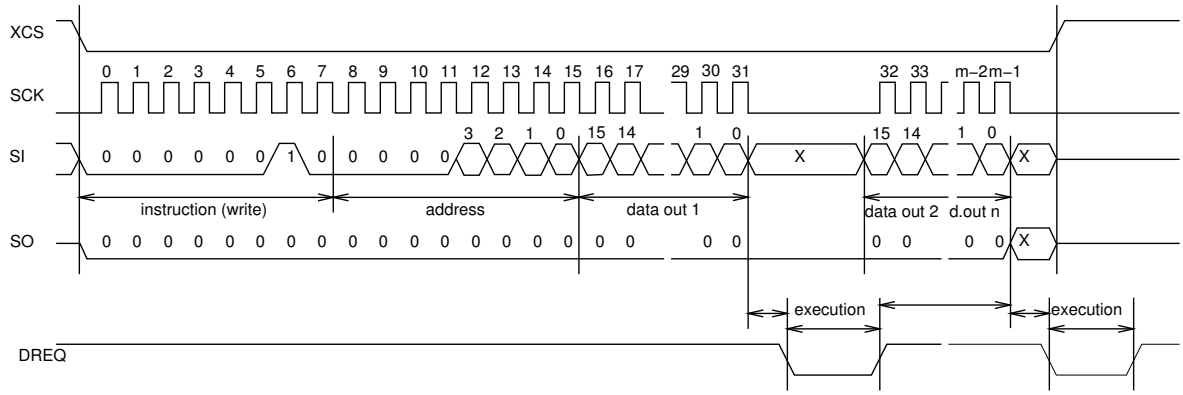


Figure 11: SCI multiple word write

VS8053b allows for the user to send multiple words to the same SCI register, which allows fast SCI uploads, shown in Figure 11. The main difference to a single write is that instead of

bringing XCS up after sending the last bit of a data word, the next data word is sent immediately. After the last data word, XCS is driven high as with a single word write.

After the last bit of a word has been sent, DREQ is driven low for the duration of the register update, marked “execution” in the figure. The time varies depending on the register and its contents (see table in Chapter 9.6 for details). If the maximum time is longer than what it takes from the microcontroller to feed the next SCI command or SDI byte, status of DREQ must be checked before finishing the next SCI/SDI operation.

7.4.4 SCI Timing Diagram

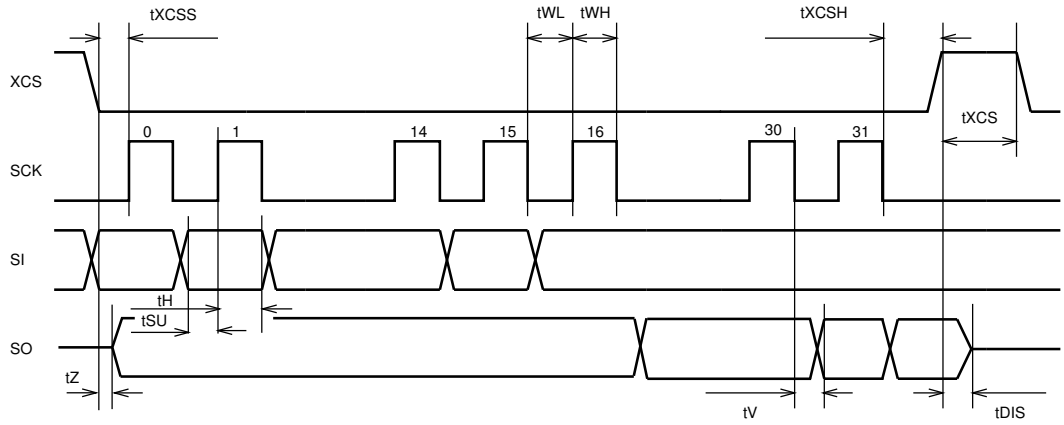


Figure 12: SPI timing diagram

The SCI timing diagram is presented in Figure 12.

Symbol	Min	Max	Unit
tXCSS	5		ns
tSU	0		ns
tH	2		CLKI cycles
tZ	0		ns
tWL	2		CLKI cycles
tWH	2		CLKI cycles
tV		2 (+ 25 ns[1])	CLKI cycles
tXCSH	1		CLKI cycles
tXCS	2		CLKI cycles
tDIS		10	ns

[1] 25 ns is when pin loaded with 100 pF capacitance. The time is shorter with lower capacitance.

Note: Although the timing is derived from the internal clock CLKI, the system always starts up in $1.0\times$ mode, thus CLKI=XTALI. After you have configured a higher clock through SCI_CLOCKF and waited for DREQ to rise, you can use a higher SPI speed as well.

Note: Because tWL + tWH + tH is $6\times$CLKI + 25 ns, the maximum speed for SCI reads is CLKI/7.

7.5 SPI Examples with SM_SDINEW and SM_SDISHARED set

7.5.1 Two SCI Writes

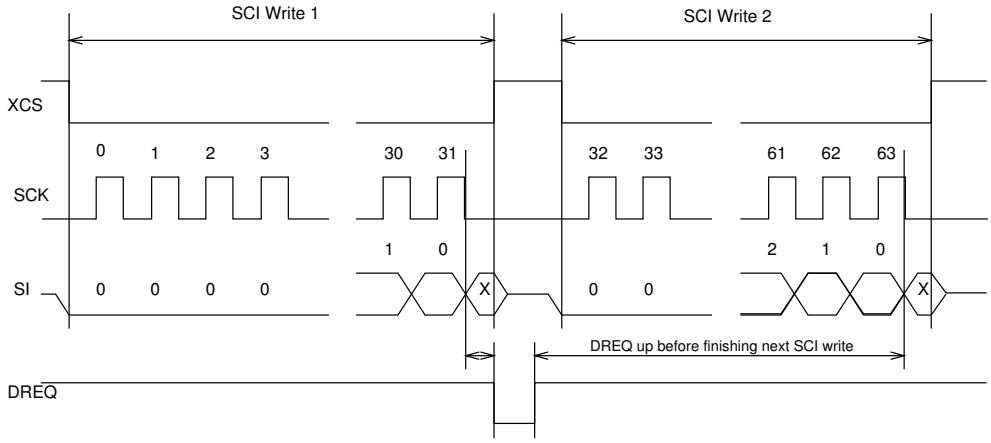


Figure 13: Two SCI operations

Figure 13 shows two consecutive SCI operations. Note that xCS *must* be raised to inactive state between the writes. Also DREQ must be respected as shown in the figure.

7.5.2 Two SDI Bytes

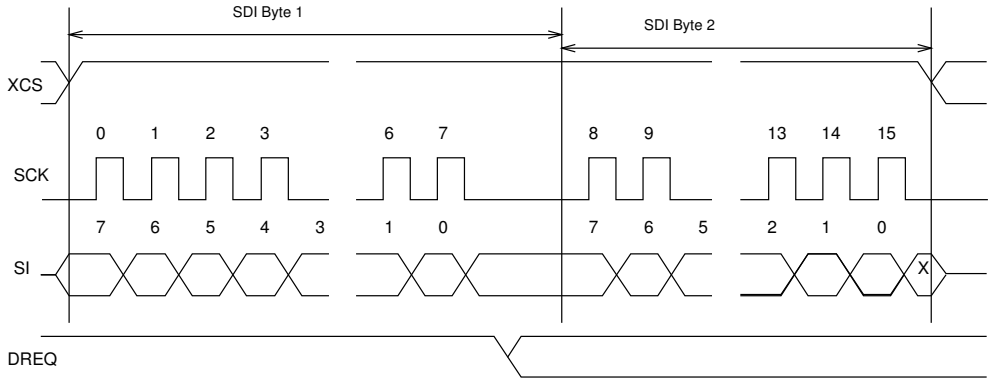


Figure 14: Two SDI bytes

SDI data is synchronized with a raising edge of xCS as shown in Figure 14. However, every byte doesn't need separate synchronization.

7.5.3 SCI Operation in Middle of Two SDI Bytes

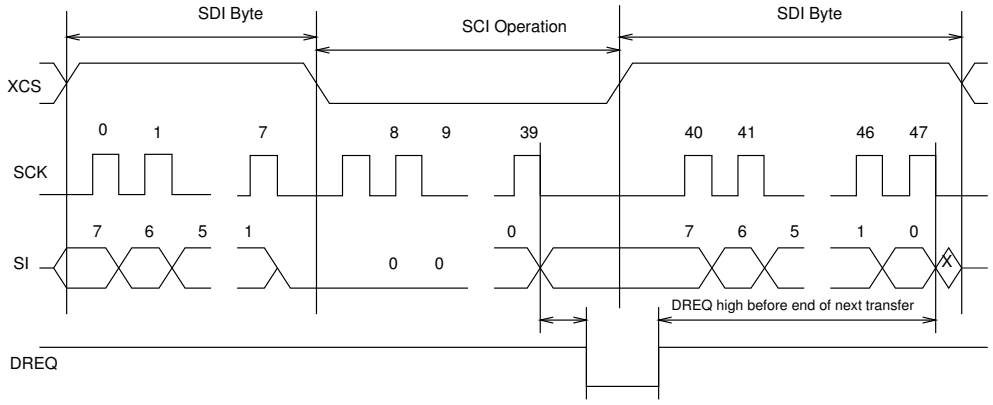


Figure 15: Two SDI bytes separated by an SCI operation

Figure 15 shows how an SCI operation is embedded in between SDI operations. xCS edges are used to synchronize both SDI and SCI. Remember to respect DREQ as shown in the figure.

8 Supported Audio Decoder Formats

Conventions	
Mark	**Description**
+	Format is supported
?	Format is supported but not thoroughly tested
-	Format exists but is not supported
	Format doesn't exist

8.1 Supported Ogg Vorbis Formats

Parameter	Min	Max	Unit
Channels		2	
Window size	64	4096	samples
Samplerate	100	48000	Hz
Bitrate		500	kbit/sec

Only floor 1 is supported. No known current encoder uses floor 0. All one- and two-channel Ogg Vorbis files should be playable with this decoder.

8.2 Supported FLAC Formats

Up to 48 kHz and 24-bit FLAC files are supported with the *VS1053b Patches w/ FLAC Decoder* plugin that is available at *http://www.vlsi.fi/en/support/software/vs10xxpatches.html* . Read the accompanying documentation of the plugin for details.

NOTE! *VS1053b Patches w/ FLAC Decoder* can also be used in the VS8053b.

8.3 Supported RIFF WAV Formats

The most common RIFF WAV subformats are supported, with 1 or 2 audio channels.

Format	Name	Supported	Comments
0x01	PCM	+	16 and 8 bits, any samplerate ≤ 48kHz
0x02	ADPCM	-	
0x03	IEEE_FLOAT	-	
0x06	ALAW	-	
0x07	MULAW	-	
0x10	OKI_ADPCM	-	
0x11	IMA_ADPCM	+	Any samplerate ≤ 48kHz
0x15	DIGISTD	-	
0x16	DIGIFIX	-	
0x30	DOLBY_AC2	-	
0x31	GSM610	-	
0x3b	ROCKWELL_ADPCM	-	
0x3c	ROCKWELL_DIGITALK	-	
0x40	G721_ADPCM	-	
0x41	G728_CELP	-	
0x50	MPEG	-	
0x55	MPEGLAYER3	-	
0x64	G726_ADPCM	-	
0x65	G722_ADPCM	-	

9 Functional Description

9.1 Main Features

VS8053b is based on a proprietary digital signal processor, VS_DSP. It contains all the code and data memory needed for Ogg Vorbis, WAV PCM + ADPCM audio decoding together with serial interfaces, a multirate stereo audio DAC and analog output amplifiers and filters. Also PCM/ADPCM audio encoding is supported using a microphone amplifier and/or line-level inputs and a stereo A/D converter. With software plugins the chip can also decode lossless FLAC as well as record the high-quality Ogg Vorbis format. A UART is provided for debugging purposes.

9.2 Data Flow of VS8053b

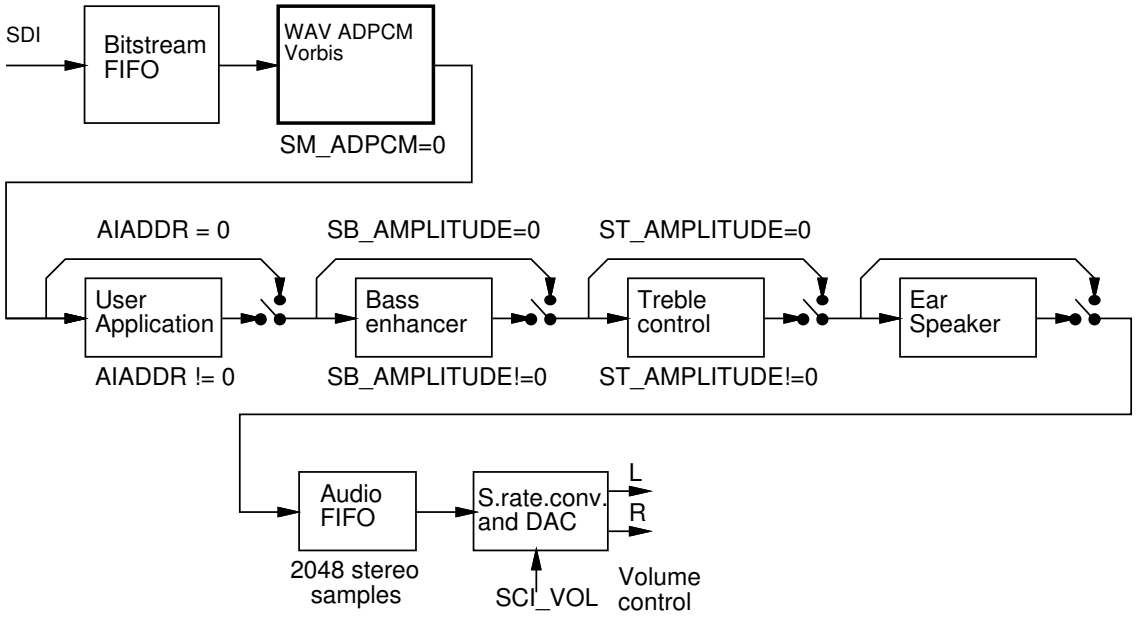


Figure 16: Data flow of VS8053b.

First, depending on the audio data, and provided ADPCM encoding mode is not set, Ogg Vorbis, PCM WAV or IMA ADPCM WAV is received and decoded from the SDI bus.

After decoding, if SCI_AIADDR is non-zero, application code is executed from the address pointed to by that register. For more details, see Application Notes for VS10XX.

Then data may be sent to the Bass Enhancer and Treble Control depending on the SCI_BASS register.

Next, headphone processing is performed, if the EarSpeaker spatial processing is active.

After that the data to the Audio FIFO, which holds the data until it is read by the Audio interrupt and fed to the samplerate converter and DACs. The size of the audio FIFO is 2048 stereo (2×16-bit) samples, or 8 KiB.

The samplerate converter upsamples all different samplerates to XTALI/2, or 128 times the highest usable samplerate with 18-bit precision. Volume control is performed in the upsampled domain. New volume settings are loaded only when the upsampled signal crosses the zero point (or after a timeout). This zero-crossing detection almost completely removes all audible noise that occurs when volume is suddenly changed.

The samplerate conversion to a common samplerate removes the need for complex PLL-based clocking schemes and allows almost unlimited sample rate accuracy with one fixed input clock frequency. With a 12.288 MHz clock, the DA converter operates at 128×48 kHz, i.e. 6.144 MHz, and creates a stereo in-phase analog signal. The oversampled output is low-pass filtered by an on-chip analog filter. This signal is then forwarded to the earphone amplifier.

9.3 EarSpeaker Spatial Processing

While listening to headphones the sound has a tendency to be localized inside the head. The sound field becomes flat and lacking the sensation of dimensions. This is an unnatural, awkward and sometimes even disturbing situation. This phenomenon is often referred in literature as 'lateralization', meaning 'in-the-head' localization. Long-term listening to lateralized sound may lead to listening fatigue.

All real-life sound sources are external, leaving traces to the acoustic wavefront that arrives to the ear drums. From these traces, the auditory system of the brain is able to judge the distance and angle of each sound source. In loudspeaker listening the sound is external and these traces are available. In headphone listening these traces are missing or ambiguous.

EarSpeaker processes sound to make listening via headphones more like listening to the same music from real loudspeakers or live music. Once EarSpeaker processing is activated, the instruments are moved from inside to the outside of the head, making it easier to separate the different instruments (see figure 17). The listening experience becomes more natural and pleasant, and the stereo image is sharper as the instruments are widely on front of the listener instead of being inside the head.

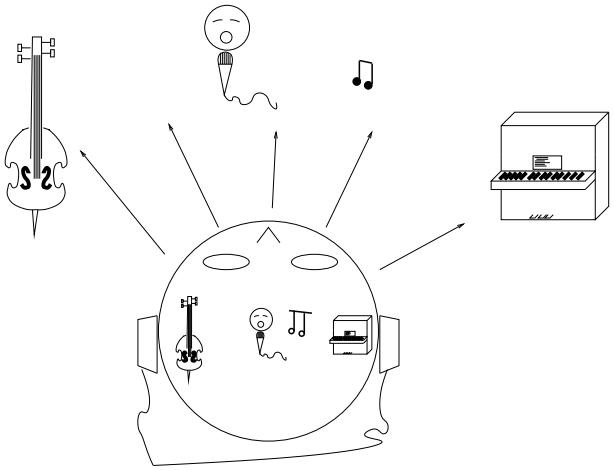


Figure 17: EarSpeaker externalized sound sources vs. normal inside-the-head sound

Note that EarSpeaker differs from any common spatial processing effects, such as echo, reverb, or bass boost. EarSpeaker accurately simulates the human auditory model and real listening environment acoustics. Thus is does not change the tonal character of the music by introducing artificial effects.

EarSpeaker processing can be parameterized to a few different modes, each simulating a little different type of acoustical situation, suiting different personal preferences and types of recording. See section 9.6.1 for how to activate different modes.

- *Off*: Best option when listening through loudspeakers or if the audio to be played contains binaural preprocessing.
- *minimal*: Suited for listening to normal musical scores with headphones, very subtle.
- *normal*: Suited for listening to normal musical scores with headphones, moves sound source further away than *minimal*.
- *extreme*: Suited for old or 'dry' recordings, or if the audio to be played is artificial, for example generated MIDI.

9.4 Serial Data Interface (SDI)

The serial data interface is meant for transferring compressed data for the different decoders of VS8053b.

If the input of the decoder is invalid or it is not received fast enough, analog outputs are automatically muted.

Also several different tests may be activated through SDI as described in Chapter 10.

9.5 Serial Control Interface (SCI)

The serial control interface is compatible with the SPI bus specification. Data transfers are always 16 bits. VS8053b is controlled by writing and reading the registers of the interface.

The main controls of the serial control interface are:

- control of the operation mode, clock, and builtin effects
- access to status information and header data
- receiving encoded data in recording mode
- uploading and controlling user programs

9.6 SCI Registers

VS8053b sets DREQ low when it detects an SCI operation (this delay is 16 to 40 CLKI cycles depending on whether an interrupt service routine is active) and restores it when it has processed the operation. The duration depends on the operation. If DREQ is low when an SCI operation is performed, it also stays low after SCI operation processing.

If DREQ is high before a SCI operation, do not start a new SCI/SDI operation before DREQ is high again. If DREQ is low before a SCI operation because the SDI can not accept more data, make certain there is enough time to complete the operation before sending another.

SCI registers, prefix SCI_					
Reg	**Type**	**Reset**	**Time**[1]	**Abbrev[bits]**	**Description**
0x0	rw	0x4000[6]	80 CLKI[4]	MODE	Mode control
0x1	rw	0x000C[3]	80 CLKI	STATUS	Status of VS8053b
0x2	rw	0	80 CLKI	BASS	Built-in bass/treble control
0x3	rw	0	1200 XTALI[5]	CLOCKF	Clock freq + multiplier
0x4	rw	0	100 CLKI	DECODE_TIME	Decode time in seconds
0x5	rw	0	450 CLKI[2]	AUDATA	Misc. audio data
0x6	rw	0	100 CLKI	WRAM	RAM write/read
0x7	rw	0	100 CLKI	WRAMADDR	Base address for RAM write/read
0x8	r	0	80 CLKI	HDAT0	Stream header data 0
0x9	r	0	80 CLKI	HDAT1	Stream header data 1
0xA	rw	0	210 CLKI[2]	AIADDR	Start address of application
0xB	rw	0	80 CLKI	VOL	Volume control
0xC	rw	0	80 CLKI[2]	AICTRL0	Application control register 0
0xD	rw	0	80 CLKI[2]	AICTRL1	Application control register 1
0xE	rw	0	80 CLKI[2]	AICTRL2	Application control register 2
0xF	rw	0	80 CLKI[2]	AICTRL3	Application control register 3

[1] This is the worst-case time that DREQ stays low after writing to this register. The user may choose to skip the DREQ check for those register writes that take less than 100 clock cycles to execute and use a fixed delay instead.

[2] In addition, the cycles spent in the user application routine must be counted.

[3] Firmware changes the value of this register immediately to 0x48 (analog enabled), and after a short while to 0x40 (analog drivers enabled).

[4] When mode register write specifies a software reset the worst-case time is 22000 XTALI cycles.

[5] If the clock multiplier is changed, writing to CLOCKF register may force internal clock to run at $1.0 \times$ XTALI for a while. Thus it is not a good idea to send SCI or SDI bits while this register update is in progress.

[6] Firmware changes the value of this register immediately to 0x4800.

Reads from all SCI registers complete in under 100 CLKI cycles, except a read from AIADDR in 200 cycles. In addition the cycles spent in the user application routine must be counted to the read time of AIADDR, AUDATA, and AICTRL0..3.

9.6.1 SCI_MODE (RW)

SCI_MODE controls the operation of VS8053b and defaults to 0x4800 (SM_SDINEW set).

Bit	Name	Function	Value	Description
0	SM_DIFF	Differential	0 1	normal in-phase audio left channel inverted
1	SM_LAYER12	Not used in VS8053	0 1	no yes
2	SM_RESET	Soft reset	0 1	no reset reset
3	SM_CANCEL	Cancel decoding current file	0 1	no yes
4	SM_EARSPEAKER_LO	EarSpeaker low setting	0 1	off active
5	SM_TESTS	Allow SDI tests	0 1	not allowed allowed
6	SM_STREAM	Stream mode	0 1	no yes
7	SM_EARSPEAKER_HI	EarSpeaker high setting	0 1	off active
8	SM_DACT	DCLK active edge	0 1	rising falling
9	SM_SDIORD	SDI bit order	0 1	MSb first MSb last
10	SM_SDISHARE	Share SPI chip select	0 1	no yes
11	SM_SDINEW	VS10xx native SPI modes	0 1	no yes
12	SM_ADPCM	PCM/ADPCM recording active	0 1	no yes
13	-	-	0 1	right wrong
14	SM_LINE1	MIC / LINE1 selector	0 1	MICP LINE1
15	SM_CLK_RANGE	Input clock range	0 1	12..13 MHz 24..26 MHz

When SM_DIFF is set, the player inverts the left channel output. For a stereo input this creates virtual surround, and for a mono input this creates a differential left/right signal.

SM_LAYER12 has no function in VS8053 because it does not support MPEG audio decoding.

Software reset is initiated by setting SM_RESET to 1. This bit is cleared automatically.

If you want to stop decoding a in the middle, set SM_CANCEL, and continue sending data honouring DREQ. When SM_CANCEL is detected by a codec, it stops decoding and return to the main loop. The stream buffer content is discarded and the SM_CANCEL bit cleared. SCI_HDAT1 is also cleared. See Chapter 10.5.2 for details.

Bits SM_EARSPEAKER_LO and SM_EARSPEAKER_HI control the EarSpeaker spatial processing. If both are 0, the processing is not active. Other combinations activate the processing and select 3 different effect levels: LO = 1, HI = 0 selects *minimal*, LO = 0, HI = 1 selects *normal*, and LO = 1, HI = 1 selects *extreme*. EarSpeaker takes approximately 12 MIPS at 44.1 kHz samplerate.

If SM_TESTS is set, SDI tests are allowed. For more details on SDI tests, look at Chapter 10.11.

SM_STREAM activates VS8053b's stream mode. In this mode, data should be sent with as even intervals as possible and preferable in blocks of less than 512 bytes, and VS8053b makes every attempt to keep its input buffer half full by changing its playback speed up to 5%. For best quality sound, the average speed error should be within 0.5%, the bitrate should not exceed 160 kbit/s and VBR should not be used. For details, see Application Notes for VS10XX. This mode only works with WAV files.

SM_DACT defines the active edge of data clock for SDI. When '0', data is read at the rising edge, when '1', data is read at the falling edge.

When SM_SDIORD is clear, bytes on SDI are sent MSb first. By setting SM_SDIORD, the user may reverse the bit order for SDI, i.e. bit 0 is received first and bit 7 last. Bytes are, however, still sent in the default order. This register bit has no effect on the SCI bus.

Setting SM_SDISHARE makes SCI and SDI share the same chip select, as explained in Chapter 7.1, if also SM_SDINEW is set.

Setting SM_SDINEW activates VS10xx native serial modes as described in Chapters 7.1.1 and 7.3.1. Note, that this bit is set as a default when VS8053b is started up.

By activating SM_ADPCM and SM_RESET at the same time, the user activates IMA ADPCM recording mode (see section 10.8).

SM_LINE_IN is used to select the left-channel input for ADPCM recording. If '0', differential microphone input pins MICP and MICN are used; if '1', line-level MICP/LINEIN1 pin is used.

SM_CLK_RANGE activates a clock divider in the XTAL input. When SM_CLK_RANGE is set, the clock is divided by 2 at the input. From the chip's point of view e.g. 24 MHz becomes 12 MHz. SM_CLK_RANGE should be set as soon as possible after a chip reset.

9.6.2 SCI_STATUS (RW)

SCI_STATUS contains information on the current status of VS8053b. It also controls some low-level things that the user does not usually have to care about.

Name	Bits	Description
SS_DO_NOT_JUMP	15	Header in decode, do not fast forward/rewind
SS_SWING	14:12	Set swing to +0 dB, +0.5 dB, .., or +3.5 dB
SS_VCM_OVERLOAD	11	GBUF overload indicator '1' = overload
SS_VCM_DISABLE	10	GBUF overload detection '1' = disable
	9:8	reserved
SS_VER	7:4	Version
SS_APDOWN2	3	Analog driver powerdown
SS_APDOWN1	2	Analog internal powerdown
SS_AD_CLOCK	1	AD clock select, '0' = 6 MHz, '1' = 3 MHz
SS_REFERENCE_SEL	0	Reference voltage selection, '0' = 1.23 V, '1' = 1.65 V

SS_DO_NOT_JUMP is set when a WAV or Ogg Vorbis header is being decoded and jumping to another location in the file is not allowed. If you use soft reset or cancel, clear this bit yourself or it can be accidentally left set.

If AVDD is at least 3.3 V, SS_REFERENCE_SEL can be set to select 1.65 V reference voltage to increase the analog output swing.

SS_AD_CLOCK can be set to divide the AD modulator frequency by 2 if XTALI/2 is too much.

SS_VER is 0 for VS1001, 1 for VS1011, 2 for VS1002, 3 for VS1003, 4 for VS1053 and VS8053, 5 for VS1033, 7 for VS1103, and 6 for VS1063.

SS_APDOWN2 controls analog driver powerdown. SS_APDOWN1 controls internal analog powerdown. These bit are meant to be used by the system firmware only.

If the user wants to powerdown VS8053b with a minimum power-off transient, set SCI_VOL to 0xffff, then wait for at least a few milliseconds before activating reset.

VS8053b contains GBUF protection circuit which disconnects the GBUF driver when too much current is drawn, indicating a short-circuit to ground. SS_VCM_OVERLOAD is high while the overload is detected. SS_VCM_DISABLE can be set to disable the protection feature.

SS_SWING allows you to go above the 0 dB volume setting. Value 0 is normal mode, 1 gives +0.5 dB, and 2 gives +1.0 dB. Settings from 3 to 7 cause the DAC modulator to be overdriven and should not be used. You can use SS_SWING with I2S to control the amount of headroom.

Note: Due to a firmware bug in the VS8053b volume calculation routine clears SS_AD_CLOCK and SS_REFERENCE_SEL bits. Writes to SCI_STATUS or SCI_VOL, and sample rate changes (if bass enhancer or treble control are active) causes the volume calculation routine to be called. See the *VS1053b Patches w/ FLAC Decoder* plugin for a workaround: *http://www.vlsi.fi/en/support/software/vs10xxpatches.html*

9.6.3 SCI_BASS (RW)

Name	Bits	Description
ST_AMPLITUDE	15:12	Treble Control in 1.5 dB steps (-8..7, 0 = off)
ST_FREQLIMIT	11:8	Lower limit frequency in 1000 Hz steps (1..15)
SB_AMPLITUDE	7:4	Bass Enhancement in 1 dB steps (0..15, 0 = off)
SB_FREQLIMIT	3:0	Lower limit frequency in 10 Hz steps (2..15)

The Bass Enhancer VSBE is a bass boosting DSP algorithm, which tries to take the most out of the users earphones without causing clipping.

VSBE is activated when SB_AMPLITUDE is non-zero. SB_AMPLITUDE should be set to the user's preferences, and SB_FREQLIMIT to roughly 1.5 times the lowest frequency the user's audio system can reproduce. For example setting SCI_BASS to 0x00f6 has 15 dB enhancement below 60 Hz. When the bass enhancer is in use with VS1053b, the bass portion is calculated in mono.

Note: Because VSBE tries to avoid clipping, it gives the best bass boost with dynamical music material, or when the playback volume is not set to maximum. It also does not create bass: the source material must have some bass to begin with.

Treble Control VSTC is activated when ST_AMPLITUDE is non-zero. For example setting SCI_BASS to 0x7a00 has 10.5 dB treble enhancement at and above 10 kHz.

Bass Enhancer uses about 2.1 MIPS and Treble Control 1.2 MIPS at 44100 Hz samplerate. Both can be on simultaneously.

In VS8053b bass and treble initialization and volume change is delayed until the next batch of samples are sent to the audio FIFO. Thus, unlike with earlier VS10XX chips, audio interrupts can no longer be missed when SCI_BASS or SCI_VOL is written to.

9.6.4 SCI_CLOCKF (RW)

The external clock multiplier SCI register SCI_CLOCKF, which has changed slightly since VS1003 and VS1033, is presented in the table below.

SCI_CLOCKF bits		
Name	**Bits**	**Description**
SC_MULT	15:13	Clock multiplier
SC_ADD	12:11	Allowed multiplier addition
SC_FREQ	10: 0	Clock frequency

SC_MULT activates the built-in clock multiplier. This multiplies XTALI to create a higher CLKI. When the multiplier is changed by more than $0.5\times$, the chip runs at $1.0\times$ clock for a few hundres clock cycles. The values are as follows:

SC_MULT	MASK	CLKI
0	0x0000	XTALI
1	0x2000	XTALI$\times 2.0$
2	0x4000	XTALI$\times 2.5$
3	0x6000	XTALI$\times 3.0$
4	0x8000	XTALI$\times 3.5$
5	0xa000	XTALI$\times 4.0$
6	0xc000	XTALI$\times 4.5$
7	0xe000	XTALI$\times 5.0$

SC_ADD tells how much the decoder firmware is allowed to add to the multiplier. It has no effect in VS8053b. Please leave it at zero.

If SC_FREQ is non-zero, it tells that the input clock XTALI is running at something else than 12.288 MHz. XTALI is set in 4 kHz steps. The formula for calculating the correct value for this register is $\frac{XTALI-8000000}{4000}$ (XTALI is in Hz).

Note: because maximum samplerate is $\frac{XTALI}{256}$, all samplerates are not available if XTALI $<$ 12.288 MHz.

9.6.5 SCI_DECODE_TIME (RW)

When decoding correct data, current decoded time is shown in this register in full seconds.

The user may change the value of this register. In that case the new value should be written twice to make absolutely certain that the change is not overwritten by the firmware. A write to SCI_DECODE_TIME also resets the `byteRate` calculation.

SCI_DECODE_TIME is reset at every hardware and software reset. It is no longer cleared when decoding of a file ends to allow the decode time to proceed automatically with looped files and with seamless playback of multiple files.

With fast playback (see the `playSpeed` extra parameter) the decode time also counts faster.

Some codecs (Ogg Vorbis) can also indicate the absolute play position, see the `positionMsec` extra parameter in section 10.10.

9.6.6 SCI_AUDATA (RW)

When decoding correct data, the current samplerate and number of channels can be found in bits 15:1 and 0 of SCI_AUDATA, respectively. Bits 15:1 contain the samplerate divided by two, and bit 0 is 0 for mono data and 1 for stereo. Writing to SCI_AUDATA changes the samplerate directly.

Example: 44100 Hz stereo data reads as 0xAC45 (44101).
Example: 11025 Hz mono data reads as 0x2B10 (11024).
Example: Writing 0xAC80 sets samplerate to 44160 Hz, stereo mode does not change.

To reduce digital power consumption when idle, you can write a low samplerate to SCI_AUDATA.

Note: Ogg Vorbis decoding overrides AUDATA change. If you want to fine-tune samplerate in streaming applications with Ogg Vorbis, use SCI_CLOCKF to control the playback rate instead of AUDATA.

9.6.7 SCI_WRAM (RW)

SCI_WRAM is used to upload application programs and data to instruction and data RAMs. The start address must be initialized by writing to SCI_WRAMADDR prior to the first write/read of SCI_WRAM. As 16 bits of data can be transferred with one SCI_WRAM write/read, and the instruction word is 32 bits long, two consecutive writes/reads are needed for each instruction word. The byte order is big-endian (i.e. most significant words first). After each full-word write/read, the internal pointer is autoincremented.

9.6.8 SCI_WRAMADDR (W)

SCI_WRAMADDR is used to set the program address for following SCI_WRAM writes/reads. Use an address offset from the following table to access X, Y, I or peripheral memory.

WRAMADDR Start...End	Dest. addr. Start...End	Bits/ Word	Description
0x1800...0x18XX	0x1800...0x18XX	16	X data RAM
0x5800...0x58XX	0x1800...0x18XX	16	Y data RAM
0x8040...0x84FF	0x0040...0x04FF	32	Instruction RAM
0xC000...0xFFFF	0xC000...0xFFFF	16	I/O

Only user areas in X, Y, and instruction memory are listed above. Other areas can be accessed, but should not be written to unless otherwise specified.

9.6.9 SCI_HDAT0 and SCI_HDAT1 (R)

For WAV files, SCI_HDAT1 contains 0x7665 (“ve”). SCI_HDAT0 contains the data rate measured in bytes per second for all supported RIFF WAVE formats: mono and stereo 8-bit or 16-bit PCM, mono and stereo IMA ADPCM. To get the bitrate of the file, multiply the value by 8.

For Ogg Vorbis files, SCI_HDAT1 contains 0x4F67 “Og”. SCI_HDAT0 contains the average data rate in bytes per second. To get the bitrate of the file, multiply the value by 8.

9.6.10 SCI_AIADDR (RW)

SCI_AIADDR indicates the start address of the application code written earlier with SCI_WRAMADDR and SCI_WRAM registers. If no application code is used, this register should not be initialized, or it should be initialized to zero. For more details, see Application Notes for VS10XX.

Note: Reading AIADDR is not recommended. It can cause samplerate to be set to a very low value.

9.6.11 SCI_VOL (RW)

SCI_VOL is a volume control for the player hardware. The most significant byte of the volume register controls the left channel volume, the low part controls the right channel volume. The channel volume sets the attenuation from the maximum volume level in 0.5 dB steps. Thus, maximum volume is 0x0000 and total silence is 0xFEFE.

Note, that after hardware reset the volume is set to full volume. Resetting the software does not reset the volume setting.

Setting SCI_VOL to 0xFFFF activates analog powerdown mode.

Example: for a volume of -2.0 dB for the left channel and -3.5 dB for the right channel: (2.0/0.5) = 4, 3.5/0.5 = 7 $\rightarrow$ SCI_VOL = 0x0407.

Example: SCI_VOL = 0x2424 $\rightarrow$ both left and right volumes are 0x24 * -0.5 = -18.0 dB.

In VS8053b bass and treble initialization and volume change is delayed until the next batch of samples are sent to the audio FIFO. Thus, audio interrupts can no longer be missed during a write to SCI_BASS or SCI_VOL.

This delays the volume setting slightly, but because the volume control is now done in the DAC hardware instead of performing it to the samples going into the audio FIFO, the overall volume change response is better than before. Also, the actual volume control has zero-cross detection, which almost completely removes all audible noise that occurs when volume is suddenly changed.

9.6.12 SCI_AICTRL[x] (RW)

SCI_AICTRL[x] registers (x=[0 .. 3]) can be used to access the user's application program.

The AICTRL registers are also used with PCM/ADPCM encoding mode.

10 Operation

10.1 Clocking

VS8053b operates on a single, nominally 12.288 MHz fundamental frequency master clock. This clock can be generated by external circuitry (connected to pin XTALI) or by the internal clock crystal interface (pins XTALI and XTALO). This clock is used by the analog parts and determines the highest available samplerate. With 12.288 MHz clock all samplerates up to 48000 Hz are available.

VS8053b can also use 24..26 MHz clocks when SM_CLK_RANGE in the SCI_MODE register is set to 1. The system clock is then divided by 2 at the clock input and the chip gets a 12..13 MHz input clock.

10.2 Hardware Reset

When the XRESET -signal is driven low, VS8053b is reset and all the control registers and internal states are set to the initial values. XRESET-signal is asynchronous to any external clock. The reset mode doubles as a full-powerdown mode, where both digital and analog parts of VS8053b are in minimum power consumption stage, and where clocks are stopped. Also XTALO is grounded.

When XRESET is asseted, all output pins go to their default states. All input pins go to high-impedance state (to input state), except SO, which is still controlled by the XCS.

After a hardware reset (or at power-up) DREQ stays down for around 22000 clock cycles, which means an approximate 1.8 ms delay if VS8053b is run at 12.288 MHz. After this the user should set such basic software registers as SCI_MODE, SCI_BASS, SCI_CLOCKF, and SCI_VOL before starting decoding. See section 9.6 for details.

If the input clock is 24..26 MHz, SM_CLK_RANGE should be set as soon as possible after a chip reset without waiting for DREQ.

Internal clock can be multiplied with a PLL. Supported multipliers through the SCI_CLOCKF register are $1.0\times \ldots 5.0\times$ the input clock. Reset value for Internal Clock Multiplier is $1.0\times$. If typical values are wanted, the Internal Clock Multiplier needs to be set to $3.5\times$ after reset. Wait until DREQ rises, then write value 0x9800 to SCI_CLOCKF (register 3). See section 9.6.4 for details.

10.3 Software Reset

In some cases the decoder software has to be reset. This is done by activating bit SM_RESET in register SCI_MODE (Chapter 9.6.1). Then wait for at least 2 μs, then look at DREQ. DREQ

stays down for about 22000 clock cycles, which means an approximate 1.8 ms delay if VS8053b is run at 12.288 MHz. After DREQ is up, you may continue playback as usual.

As opposed to all earlier VS10XX chips, it is not recommended to do a software reset between songs. This way the user may be sure that even files with low samplerates or bitrates are played right to their end.

10.4 Low Power Mode

If you need to keep the system running while not decoding data, but need to lower the power consumption, you can use the following tricks.

- Select the $1.0\times$ clock by writing 0x0000 to SCI_CLOCKF. This disables the PLL and saves some power.
- Write a low non-zero value, such as 0x0010 to SCI_AUDATA. This reduces the samplerate and the number of audio interrupts required. Between audio interrupts the VSDSP core just waits for an interrupt, thus saving power.
- Turn off all audio post-processing (tone controls and EarSpeaker).
- If possible for the application, write 0xffff to SCI_VOL to disable the analog drivers.

To return from low-power mode, revert register values in reverse order.

Note: The low power mode consumes significantly more electricity than hardware reset.

10.5 Play and Decode

This is the normal operation mode of VS8053b. SDI data is decoded. Decoded samples are converted to analog domain by the internal DAC. If no decodable data is found, SCI_HDAT0 and SCI_HDAT1 are set to 0.

When there is no input for decoding, VS8053b goes into idle mode (lower power consumption than during decoding) and actively monitors the serial data input for valid data.

10.5.1 Playing a Whole File

This is the default playback mode.

1. Send an audio file to VS8053b.
2. Read extra parameter value endFillByte (Chapter 10.10).
3. Send at least 2052 bytes of endFillByte[7:0].
4. Set SCI_MODE bit SM_CANCEL.
5. Send at least 32 bytes of endFillByte[7:0].
6. Read SCI_MODE. If SM_CANCEL is still set, go to 5. If SM_CANCEL hasn't cleared after sending 2048 bytes, do a software reset (this should be extremely rare).
7. The song has now been successfully sent. HDAT0 and HDAT1 should now both contain 0 to indicate that no format is being decoded. Return to 1.

10.5.2 Cancelling Playback

Cancelling playback of a song is a normal operation when the user wants to jump to another song while doing playback.

1. Send a portion of an audio file to VS8053b.
2. Set SCI_MODE bit SM_CANCEL.
3. Continue sending audio file, but check SM_CANCEL after every 32 bytes of data. If it is still set, goto 3. If SM_CANCEL doesn't clear after 2048 bytes or one second, do a software reset (this should be extremely rare).
4. When SM_CANCEL has cleared, read extra parameter value endFillByte (Chapter 10.10).
5. Send 2052 bytes of endFillByte[7:0].
6. HDAT0 and HDAT1 should now both contain 0 to indicate that no format is being decoded. You can now send the next audio file.

10.5.3 Fast Play

VS8053b allows fast audio playback. If your microcontroller can feed data fast enough to the VS8053b, this is the preferred way to fast forward audio.

1. Start sending an audio file to VS8053b.
2. To set fast play, set extra parameter value playSpeed (Chapter 10.10).
3. Continue sending audio file.
4. To exit fast play mode, write 1 to playSpeed.

To estimate whether or not your microcontroller can feed enough data to VS8053b in fast play mode, see contents of extra parameter value byteRate (Chapter 10.10). Note that byteRate contains the data speed of the file played back at nominal speed even when fast play is active.

Note: Play speed is not reset when song is changed.

10.5.4 Fast Forward and Rewind without Audio

To do fast forward and rewind you need the capability to do random access to the audio file. Unfortunately fast forward and rewind isn't available at all times, like when file headers are being read.

1. Send a portion of an audio file to VS8053b.
2. When random access is required, read SCI_STATUS bit SS_DO_NOT_JUMP. If that bit is set, random access cannot be performed, so go back to 1.
3. Read extra parameter value endFillByte (Chapter 10.10).
4. Send at least 2048 bytes of endFillByte[7:0].
5. Jump forwards or backwards in the file.
6. Continue sending the file.

Note: It is recommended that playback volume is decreased by e.g. 10 dB when fast forwarding/rewinding.

Note: Register DECODE_TIME does not take jumps into account.

10.5.5 Maintaining Correct Decode Time

When fast forward and rewind operations are performed, there is no way to maintain correct decode time for most files. However, Ogg Vorbis files offer exact time information in the file. To use accurate time information whenever possible, use the following algorithm:

1. Start sending an audio file to VS8053b.
2. Read extra parameter value pair positionMsec (Chapter 10.10).
3. If positionMsec is -1, show you estimation of decoding time using DECODE_TIME (and your estimate of file position if you have performed fast forward / rewind operations).
4. If positionMsec is not -1, use this time to show the exact position in the file.

10.6 Feeding PCM Data

VS8053b can be used as a PCM decoder by sending a WAV file header. If the length sent in the WAV header is 0xFFFFFFFF, VS8053b stays in PCM mode indefinitely (or until SM_CANCEL has been set). 8-bit (unsigned) linear and 16-bit (signed, 2's complement) linear audio is supported in mono or stereo. A WAV header looks like this:

File Offset	Field Name	Size	Bytes	Description
0	ChunkID	4	`"RIFF"`	
4	ChunkSize	4	0xff 0xff 0xff 0xff	
8	Format	4	`"WAVE"`	
12	SubChunk1ID	4	`"fmt "`	
16	SubChunk1Size	4	0x10 0x0 0x0 0x0	16
20	AudioFormat	2	0x1 0x0	Linear PCM
22	NumOfChannels	2	C0 C1	1 for mono, 2 for stereo
24	SampleRate	4	S0 S1 S2 S3	0x1f40 for 8 kHz
28	ByteRate	4	R0 R1 R2 R3	0x3e80 for 8 kHz 16-bit mono
32	BlockAlign	2	A0 A1	0x02 0x00 for mono, 0x04 0x00 for stereo 16-bit
34	BitsPerSample	2	B0 B1	0x10 0x00 for 16-bit data
52	SubChunk2ID	4	`"data"`	
56	SubChunk2Size	4	0xff 0xff 0xff 0xff	Data size

The rules to calculate the four variables are as follows:

- $S =$ sample rate in Hz, e.g. 44100 for 44.1 kHz.
- For 8-bit data $B = 8$, and for 16-bit data $B = 16$.
- For mono data $C = 1$, for stereo data $C = 2$.
- $A = \frac{C \times B}{8}$.
- $R = S \times A$.

Example: A 44100 Hz 16-bit stereo PCM header would read as follows:

```
0000  52 49 46 46 ff ff ff ff  57 41 56 45 66 6d 74 20  |RIFF....WAVEfmt |
0100  10 00 00 00 01 00 02 00  44 ac 00 00 10 b1 02 00  |........D.......|
0200  04 00 10 00 64 61 74 61  ff ff ff ff              |....data....|
```

10.7 Ogg Vorbis Recording

Ogg Vorbis is an open file format that allows for very high sound quality with low to medium bitrates.

Ogg Vorbis recording is activated by loading the Ogg Vorbis Encoder Application to the 16 KiB program RAM memory of the VS8053b. After activation, encoder results can be read from registers SCI_HDAT0 and SCI_HDAT1, much like when using PCM/ADPCM recording (Chapter 10.8).

Three profiles are provided: one for high-quality stereo recording at a bitrate of approx. 140 kbit/s, and two for speech-quality mono recording at a bitrates between 15 and 30 kbit/s.

To use the Ogg Vorbis Encoder application, please load the application from VLSI Solution's Web page *http://www.vlsi.fi/en/support/software/vs10xxapplications.html* and read the accompanying documentation.

10.8 PCM / ADPCM Recording

This chapter explains how to record a RIFF/WAV file in PCM or IMA ADPCM format. IME ADPCM is a widely supported ADPCM format and many PC audio playback programs can play it. IMA ADPCM recording gives a compression ratio of almost 4:1 compared to linear, 16-bit audio. This makes it possible to record for example ono 8 kHz audio at 32.44 kbit/s.

VS8053 has a stereo ADC, thus also two-channel (separate AGC, if AGC enabled) and stereo (common AGC, if AGC enabled) modes are available. Mono recording mode selects either the left or right channel. Left channel is either MIC or LINE1 depending on the SCI_MODE register.

If absolute best quality stereo PCM recording at 48 kHz is required, download and use the VS1053 WAV PCM Recorder Application plugin, available for download at *http://www.vlsi.fi/en/support/software/vs10xxapplications.html* . If you use it, follow the instructions of the plugin documentation instead of this datasheet.

10.8.1 Activating PCM / ADPCM Recording Mode

Register	Bits	Description
SCI_MODE	2, 12, 14	Start ADPCM mode, select MIC/LINE1
SCI_AICTRL0	15..0	Sample rate 8000..48000 Hz (read at recording startup)
SCI_AICTRL1	15..0	Recording gain (1024 = 1×) or 0 for automatic gain control
SCI_AICTRL2	15..0	Maximum autogain amplification (1024 = 1×, 65535 = 64×)
SCI_AICTRL3	1..0	0 = joint stereo (common AGC), 1 = dual channel (separate AGC), 2 = left channel, 3 = right channel
	2	0 = IMA ADPCM mode, 1 = LINEAR PCM mode
	15..3	reserved, set to 0

PCM / IMA ADPCM recording mode is activated by setting bit SM_ADPCM in SCI_MODE and loading and starting a patch code. Line input 1 is used instead of differential mic input if SM_LINE1 is set. Before activating ADPCM recording, user **must** write the right values to SCI_AICTRL0 and SCI_AICTRL3. These values are only read at recording startup. SCI_AICTRL1 and SCI_AICTRL2 can be altered anytime, but it is preferable to write good init values before activation.

SCI_AICTRL1 controls linear recording gain. 1024 is equal to digital gain 1, 512 is equal to digital gain 0.5 and so on. If the user wants to use automatic gain control (AGC), SCI_AICTRL1 should be set to 0. Typical speech applications usually are better off using AGC, as this takes care of relatively uniform speech loudness in recordings.

SCI_AICTRL2 controls the maximum AGC gain. This can be used to limit the amplification of noise when there is no signal. If SCI_AICTRL2 is zero, the maximum gain is initialized to 65535 (64×), i.e. whole range is used.

For example:

```
  WriteVS10xxRegister(SCI_AICTRL0, 16000U);
  WriteVS10xxRegister(SCI_AICTRL1, 0);
  WriteVS10xxRegister(SCI_AICTRL2, 4096U);
  WriteVS10xxRegister(SCI_AICTRL3, 0);
  WriteVS10xxRegister(SCI_MODE, ReadVS10xxRegister(SCI_MODE) |
                                SM_ADPCM | SM_LINE1);
#ifdef I_HAVE_THE_VS1053B_PATCHES_PACKAGE
 /* Strongly recommended to use the VS1053b Patches package. Get it at
    http://www.vlsi.fi/en/support/software/vs10xxpatches.html */
  LoadVS1053PatchesPackage(); /* Loads patches and starts recording */
#else
  /* If not using the VS1053b Patches package, you need to do these steps */
  WriteVS10xxPatch();
#endif
```

selects 16 kHz, stereo mode with automatic gain control and maximum amplification of $4\times$.

It is strongly recommended to always run the VS1053b Patches package, available at *http://www.vlsi.fi/en/support/software/vs10xxpatches.html* , when encoding. If you are not running the VS1053b Patches package, you need to implement WriteVS10xxPatch() for the example above. The function must perform the following SCI writes:

Register	Reg. No	Values
SCI_WRAMADDR	0x7	0x8050
SCI_WRAM	0x6	0x0000,0x1790,0xf400,0x5400,0x0000,0x0a10,0xf400,0x5600, 0xb080,0x0024,0x0007,0x9257,0x3f00,0x0024,0x0030,0x0297, 0x3f00,0x0024,0x0000,0x004d,0x0014,0x958f,0x0000,0x1b4e, 0x280f,0xe100,0x0006,0x2016,0x2a00,0x17ce,0x3e12,0xb817, 0x3e14,0xf812,0x3e01,0xb811,0x0007,0x9717,0x0020,0xffd2, 0x0030,0x11d1,0x3111,0x8024,0x3704,0xc024,0x3b81,0x8024, 0x3101,0x8024,0x3b81,0x8024,0x3f04,0xc024,0x2808,0x4800, 0x36f1,0x9811,0x2814,0x9c91,0x0000,0x004d,0x2814,0x9940, 0x003f,0x0013,
SCI_AIADDR	0xa	0x0050

This small and incomplete patch is also available from VLSI Solution's web page *http://www.vlsi.fi/en/support/software/vs10xxpatches.html* by the name of *VS1053b IMA ADPCM Encoder Fix*.

10.8.2 Reading PCM / IMA ADPCM Data

After PCM / IMA ADPCM recording has been activated, registers SCI_HDAT0 and SCI_HDAT1 have new functions.

The PCM / IMA ADPCM sample buffer is 1024 16-bit words. The fill status of the buffer can be read from SCI_HDAT1. If SCI_HDAT1 is greater than 0, you can read as many 16-bit words from SCI_HDAT0. If the data is not read fast enough, the buffer overflows and returns to empty state.

Note: if SCI_HDAT1 ≥ 768, it may be better to wait for the buffer to overflow and clear before reading samples. That way you may avoid buffer aliasing.

In IMA ADPCM mode each mono IMA ADPCM block is 128 words, i.e. 256 bytes, and stereo IMA ADPCM block is 256 words, i.e. 512 bytes. If you wish to interrupt reading data and possibly continue later, please stop at the boundary. This way complete compression blocks are skipped and the encoded stream stays valid.

10.8.3 Adding a PCM RIFF Header

To make your PCM file a RIFF / WAV file, you have to add a header to the data. The following shows a header for a mono file. Note that 2- and 4-byte values are little-endian (lowest byte first).

File Offset	Field Name	Size	Bytes	Description
0	ChunkID	4	"RIFF"	
4	ChunkSize	4	F0 F1 F2 F3	File size - 8
8	Format	4	"WAVE"	
12	SubChunk1ID	4	"fmt "	
16	SubChunk1Size	4	0x10 0x0 0x0 0x0	20
20	AudioFormat	2	0x01 0x0	0x1 for PCM
22	NumOfChannels	2	C0 C1	1 for mono, 2 for stereo
24	SampleRate	4	R0 R1 R2 R3	0x1f40 for 8 kHz
28	ByteRate	4	B0 B1 B2 B3	0x3e80 for 8 kHz mono
32	BlockAlign	2	0x02 0x00	2 for mono, 4 for stereo
34	BitsPerSample	2	0x10 0x00	16 bits / sample
36	SubChunk3ID	4	"data"	
40	SubChunk3Size	4	D0 D1 D2 D3	Data size (File Size-36)
44	Samples...			Audio samples

The values in the table are calculated as follows:
$R = F_s$ (see Chapter 10.8.1 to see how to calculate F_s)
$B = 2 \times F_s \times C$

If you know beforehand how much you are going to record, you may fill in the complete header before any actual data. However, if you don't know how much you are going to record, you have to fill in the header size datas F and D after finishing recording.

The PCM data is read from SCI_HDAT0 and written into file as follows. The low 8 bits of SCI_HDAT0 should be written as the first byte to a file, then the high 8 bits (little-endian order). Note that this is different from how ADPCM data is written to a file (see Chapter 10.8.4).

Below is an example of a valid header for a 44.1 kHz mono PCM file that has a final length of 1798768 (0x1B7270) bytes:

```
0000  52 49 46 46 68 72 1b 00  57 41 56 45 66 6d 74 20  |RIFFhr..WAVEfmt |
0010  10 00 00 00 01 00 01 00  80 bb 00 00 00 77 01 00  |.............w..|
0020  02 00 10 00 64 61 74 61  44 72 1b 00              |....dataDr......|
```

10.8.4 Adding an IMA ADPCM RIFF Header

To make your IMA ADPCM file a RIFF / WAV file, you have to add a header to the data. The following shows a header for a mono file. Note that 2- and 4-byte values are little-endian (lowest byte first).

File Offset	Field Name	Size	Bytes	Description
0	ChunkID	4	`"RIFF"`	
4	ChunkSize	4	F0 F1 F2 F3	File size - 8
8	Format	4	`"WAVE"`	
12	SubChunk1ID	4	`"fmt "`	
16	SubChunk1Size	4	0x14 0x0 0x0 0x0	20
20	AudioFormat	2	0x11 0x0	0x11 for IMA ADPCM
22	NumOfChannels	2	C0 C1	1 for mono, 2 for stereo
24	SampleRate	4	R0 R1 R2 R3	0x1f40 for 8 kHz
28	ByteRate	4	B0 B1 B2 B3	0xfd7 for 8 kHz mono
32	BlockAlign	2	0x00 0x01	256 for mono, 512 for stereo
34	BitsPerSample	2	0x04 0x00	4-bit ADPCM
36	ByteExtraData	2	0x02 0x00	2
38	ExtraData	2	0xf9 0x01	Samples per block (505)
40	SubChunk2ID	4	`"fact"`	
44	SubChunk2Size	4	0x4 0x0 0x0 0x0	4
48	NumOfSamples	4	S0 S1 S2 S3	
52	SubChunk3ID	4	`"data"`	
56	SubChunk3Size	4	D0 D1 D2 D3	Data size (File Size-60)
60	Block1	256		First ADPCM block, 512 bytes for stereo
316	...			More ADPCM data blocks

If we have n audio blocks, the values in the table are as follows:
$F = n \times C \times 256 + 52$
$R = F_s$ (see Chapter 10.8.1 to see how to calculate F_s)
$B = \frac{F_s \times C \times 256}{505}$
$S = n \times 505$. $D = n \times C \times 256$

If you know beforehand how much you are going to record, you may fill in the complete header before any actual data. However, if you don't know how much you are going to record, you have to fill in the header size datas F, S and D after finishing recording.

The 128 words (256 words for stereo) of an ADPCM block are read from SCI_HDAT0 and written into file as follows. The high 8 bits of SCI_HDAT0 should be written as the first byte to a file, then the low 8 bits (big-endian order). Note that this is contrary to the native byte order of some 16-bit microcontrollers, and you may have to take extra care to do this right. Note also, that this is different from how PCM data is written to a file (see Chapter 10.8.3).

To see if you have written the mono file in the right way check bytes 2 and 3 (the first byte counts as byte 0) of each 256-byte block. Byte 2 should be in the range $0..88$ and byte 3 should be zero. For stereo you check bytes 2, 3, 6, and 7 of each 512-byte block. Bytes 2 and 6 should be in the range $0..88$. Bytes 3 and 7 should be zero.

Below is an example of a valid header for a 44.1 kHz stereo IMA ADPCM file that has a final

length of 10038844 (0x992E3C) bytes:

```
0000  52 49 46 46 34 2e 99 00  57 41 56 45 66 6d 74 20  |RIFF4...WAVEfmt |
0010  14 00 00 00 11 00 02 00  44 ac 00 00 a7 ae 00 00  |........D.......|
0020  00 02 04 00 02 00 f9 01  66 61 63 74 04 00 00 00  |........fact....|
0030  14 15 97 00 64 61 74 61  00 2e 99 00              |....data....|
```

10.8.5 Playing ADPCM Data

In order to play back your PCM / IMA ADPCM recordings, you have to have a file with a header as described in Chapter 10.8.3 or Chapter 10.8.4. If this is the case, all you need to do is to provide the ADPCM file through SDI as you would with any audio file.

10.8.6 Sample Rate Considerations

VS10xx chips that support IMA ADPCM playback are capable of playing back ADPCM files with any sample rate. However, some other programs may expect IMA ADPCM files to have some exact sample rates, like 8000 or 11025 Hz. Also, some programs or systems do not support sample rates below 8000 Hz.

If you want better quality with the expense of increased data rate, you can use higher sample rates, for example 16 kHz.

10.8.7 Record Monitoring Volume

In VS8053b writing to the SCI_VOL register during IMA ADPCM encoding does not change the volume. You need to set a suitable volume before activating the IMA ADPCM mode, or you can use the VS8053 hardware volume control register DAC_VOL directly.

For example:

```
WriteVS10xxRegister(SCI_WRAMADDR, 0xc045);    /*DAC_VOL*/
WriteVS10xxRegister(SCI_WRAM,     0x0101);    /*-6.0 dB*/
```

The hardware volume control DAC_VOL (address 0xc045) allows 0.5 dB steps for both left (high 8 bits) and right channel (low 8 bits). The low 4 bits of both 8-bit values set the attenuation in 6 dB steps (range $0..15$), the high 4 bits in 0.5 dB steps (range $0..11$). For examples, see table below.

dB	DAC_VOL	dB	DAC_VOL	dB	DAC_VOL	dB	DAC_VOL
-0.0	0x0000	-6.5	0xb2b2	:	:	-60.0	0x0a0a
-0.5	0xb1b1	:	:	-36.0	0x0606	-60.5	0xbbbb
-1.0	0xa1a1	-12.0	0x0202	-36.5	0xb7b7	:	:
-1.5	0x9191	-12.5	0xb3b3	:	:	-66.0	0x0b0b
-2.0	0x8181	:	:	-42.0	0x0707	-66.5	0xbcbc
-2.5	0x7171	-18.0	0x0303	-42.5	0xb8b8	:	:
-3.0	0x6161	-18.5	0xb4b4	:	:	-72.0	0x0c0c
-3.5	0x5151	:	:	-48.0	0x0808	-72.5	0xbdbd
-4.0	0x4141	-24.0	0x0404	-48.5	0xb9b9	:	:
-4.5	0x3131	-24.5	0xb5b5	:	:	-78.0	0x0d0d
-5.0	0x2121	:	:	-54.0	0x0909	-78.5	0xbebe
-5.5	0x1111	-30.0	0x0505	-54.5	0xbaba	:	:
-6.0	0x0101	-30.5	0xb6b6	:	:	-84.0	0x0e0e

10.9 SPI Boot

If GPIO0 is set with a pull-up resistor to IOVDD at boot time, VS8053b tries to boot from an external SPI memory. If you have GPIO0 pulled low, also pull GPIO1 low, unless you want to start the real-time MIDI mode.

The SPI boot redefines the following pins:

Normal Mode	SPI Boot Mode
GPIO0	xCS
GPIO1	CLK
DREQ	MOSI
GPIO2	MISO

The SPI memory has to be an SPI Bus Serial EEPROM or FLASH with 16-bit or 24-bit addresses. The serial speed used by VS8053b is 245 kHz with the nominal 12.288 MHz clock. The first three bytes in the memory have to be 0x50, 0x26, 0x48.

10.10 Extra Parameters

The following structure is in X memory at address 0x1e02 (note the different location than in VS1033) and can be used to change some extra parameters or get useful information.

```
#define PARAMETRIC_VERSION 0x0003
struct parametric {
  /* configs are not cleared between files */
  u_int16 version;    /*1e02 - structure version */
  u_int16 config1;    /*1e03 ---- ---- ppss RRRR  PS mode, SBR mode, Reverb */
  u_int16 playSpeed;  /*1e04 0,1 = normal speed, 2 = twice, 3 = three times etc. */
  u_int16 byteRate;   /*1e05 average byterate */

  u_int16 endFillByte;    /*1e06 byte value to send after file sent */
  u_int16 reserved[16];   /*1e07..15 file byte offsets */
  u_int32 jumpPoints[8];  /*1e16..25 file byte offsets */
  u_int16 latestJump;  /*1e26 index to lastly updated jumpPoint */
  u_int32 positionMsec /*1e27-28 play position, if known (WMA, Ogg Vorbis) */
  s_int16 resync;      /*1e29 > 0 for automatic m4a, ADIF, WMA resyncs */
  union {
    struct {
      u_int32 curPacketSize;
      u_int32 packetSize;
    } wma;
    struct {
      u_int16 sceFoundMask; /*1e2a SCE's found since last clear */
      u_int16 cpeFoundMask; /*1e2b CPE's found since last clear */
      u_int16 lfeFoundMask; /*1e2c LFE's found since last clear */
      u_int16 playSelect;   /*1e2d 0 = first any, initialized at aac init */
      s_int16 dynCompress;  /*1e2e -8192=1.0, initialized at aac init */
      s_int16 dynBoost;     /*1e2f  8192=1.0, initialized at aac init */
      u_int16 sbrAndPsStatus; /*0x1e30 1=SBR, 2=upsample, 4=PS, 8=PS active */
    } aac;
    struct {
      u_int32 bytesLeft;
    } midi;
    struct {
      s_int16 gain; /* 0x1e2a proposed gain offset in 0.5dB steps, default = -12 */
    } vorbis;
  } i;
};
```

Notice that reading two-word variables through the SCI_WRAMADDR and SCI_WRAM interface is not protected in any way. The variable can be updated between the read of the low and high parts. The problem arises when both the low and high parts change values. To determine if the value is correct, you should read the value twice and compare the results.

The following example shows what happens when `bytesLeft` is decreased from 0x10000 to 0xffff and the update happens between low and high part reads or after high part read.

Read Invalid		Read Valid		No Update	
Address	**Value**	**Address**	**Value**	**Address**	**Value**
0x1e2a	0x0000 change after this	0x1e2a	0x0000	0x1e2a	0x0000
0x1e2b	0x0000	0x1e2b	0x0001 change after this	0x1e2b	0x0001
0x1e2a	0xffff	0x1e2a	0xffff	0x1e2a	0x0000
0x1e2b	0x0000	0x1e2b	0x0000	0x1e2b	0x0001

You can see that in the invalid read the low part wraps from 0x0000 to 0xffff while the high part

stays the same. In this case the second read gives a valid answer, otherwise always use the value of the first read. The second read is needed when it is possible that the low part wraps around, changing the high part, i.e. when the low part is small. `bytesLeft` is only decreased by one at a time, so a reread is needed only if the low part is 0.

10.10.1 Common Parameters

These parameters are common for all codecs. Other fields are only valid when the corresponding codec is active. The currently active codec can be determined from SCI_HDAT1.

Parameter	Address	Usage
chipID	0x1e00-01	Fuse-programmed ID (cosmetic copy of the fuses)
version	0x1e02	Structure version – 0x0003
config1	0x1e03	Miscellaneous configuration
playSpeed	0x1e04	0,1 = normal speed, 2 = twice, 3 = three times etc.
byteRate	0x1e05	average byterate
endFillByte	0x1e06	byte to send after file
jumpPoints[8]	0x1e16-25	Not used
latestJump	0x1e26	Index to latest jumpPoint
positionMsec	0x1e27-28	File position in milliseconds, if available
resync	0x1e29	Not used

The fuse-programmed ID is read at startup and copied into the `chipID` field. If not available, the value is all zeros.

The `version` field can be used to determine the layout of the rest of the structure. The version number is changed when the structure is changed. For VS8053b the structure version is 3.

`config1` isn't used in VS8053.

`playSpeed` makes it possible to fast forward songs. Decoding of the bitstream is performed, but only each `playSpeed` frames are played. For example by writing 4 to `playSpeed` plays the song four times as fast as normal, if you are able to feed the data with that speed. Write 0 or 1 to return to normal speed. SCI_DECODE_TIME also counts faster. All current codecs support the `playSpeed` configuration.

`byteRate` contains the average bitrate in bytes per second for every code. The value is updated once per second and it can be used to calculate an estimate of the remaining playtime. This value is also available in SCI_HDAT0 for all codecs.

`endFillByte` indicates what byte value to send after file is sent before SM_CANCEL.

`positionMsec` is a field that gives the current play position in a file in milliseconds, regardless of rewind and fast forward operations. The value is only available in codecs that can determine the play position from the stream itself. Currently Ogg Vorbis provide this information. If the

position is unknown, this field contains -1.

The file size and sample size information of WAV files are ignored when `resync` is non-zero. The user must use SM_CANCEL or software reset to end decoding.

10.10.2 Ogg Vorbis

Parameter	Address	Usage
gain	0x1e2a	Preferred Replay Gain offset

Ogg Vorbis decoding supports Replay Gain technology. The Replay Gain technology is used to automatically give all songs a matching volume so that the user does not need to adjust the volume setting between songs.

If the Ogg Vorbis decoder finds a REPLAYGAIN_ALBUM_GAIN tag in the song header, the tag is parsed and the decoded gain setting is written to the `gain` parameter.
If REPLAYGAIN_ALBUM_GAIN is not available, REPLAYGAIN_TRACK_GAIN is used.
If even REPLAYGAIN_TRACK_GAIN is not available, a default of -6 dB (`gain` value -12) is set.

For more information about Replay Gain, see *http://en.wikipedia.org/wiki/Replay_Gain* and *http://www.replaygain.org/* .

The player software can use the gain value to adjust the volume level. Negative values mean that the volume should be decreased, positive values mean that the volume should be increased.

For example `gain` = -11 means that volume should be decreased by 5.5 dB ($-11/2 = -5.5$), and left and right attenuation should be increased by 11. When `gain` = 2 volume should be increased by 1 dB ($2/2 = 1.0$), and left and right attenuation should be decreased by 2. Because volume setting can not go above +0 dB, the value should be saturated.

Gain	Volume	SCI_VOL (Volume-Gain)
-11 (-5.5 dB)	0 (+0.0 dB)	0x0b0b (-5.5 dB)
-11 (-5.5 dB)	3 (-1.5 dB)	0x0e0e (-7.0 dB)
+2 (+1.0 dB)	0 (+0.0 dB)	0x0000 (+0.0 dB)
+2 (+1.0 dB)	1 (-0.5 dB)	0x0000 (+0.0 dB)
+2 (+1.0 dB)	4 (-2.0 dB)	0x0202 (-1.0 dB)

10.11 SDI Tests

There are several test modes in VS8053b, which allow the user to perform memory tests, SCI bus tests, and several different sine wave tests.

All tests, except for the New Sine and Sweep Tests, are started in a similar way: do a hardware reset to VS8053b, then set register SM_MODE bit SM_TESTS, and then send a test command sequence to the SDI bus. Each test is started by sending a 4-byte special command sequence, followed by 12 zeros. The sequences are described below.

10.11.1 Old Sine Test

Sine test is initialized with the 16-byte sequence 0x53 0xEF 0x6E n 0 0 0 0 0 0 0 0 0 0 0 0, where n defines the sine test to use. n is defined as follows:

n bits		
Name	**Bits**	**Description**
F_sIdx	7:5	Samplerate index
S	4:0	Sine skip speed

F_sIdx	F_s	F_sIdx	F_s
0	44100 Hz	4	24000 Hz
1	48000 Hz	5	16000 Hz
2	32000 Hz	6	11025 Hz
3	22050 Hz	7	12000 Hz

The frequency of the sine to be output can now be calculated from $F = F_s \times \frac{S}{128}$.

Example: Sine test is activated with value 126, which is 0b01111110. Breaking n to its components, $F_sIdx = 0b011 = 3$ and thus $F_s = 22050Hz$. $S = 0b11110 = 30$, and thus the final sine frequency $F = 22050Hz \times \frac{30}{128} \approx 5168Hz$.

To exit the sine test, send the sequence 0x45 0x78 0x69 0x74 0 0 0 0 0 0 0 0 0 0 0 0.

Note: Sine test signals go through the digital volume control, so it is possible to test channels separately. The samplerate can also be changed during the test by writing the new rate to SCI_AUDATA.

10.11.2 New Sine and Sweep Tests

A more frequency-accurate sine test can be started and controlled from SCI. SCI_AICTRL0 and SCI_AICTRL1 set the sine frequencies for left and right channel, respectively. These registers, volume (SCI_VOL), and samplerate (SCI_AUDATA) can be set before or during the test. Write 0x4020 to SCI_AIADDR to start the test.

SCI_AICTRLn can be calculated from the desired frequency and DAC samplerate by:

$$SCI\_AICTRLn = F_{sin} \times 65536/F_s$$

The maximum value for SCI_AICTRLn is 0x8000U. For the best S/N ratio for the generated sine, three LSb's of the SCI_AICTRLn should be zero. The resulting frequencies F_{sin} can be calculated from the DAC samplerate F_s and SCI_AICTRL0 / SCI_AICTRL1 using the following equation.

$$F_{sin} = SCI\_AICTRLn \times F_s/65536$$

Sine sweep test can be started by writing 0x4022 to SCI_AIADDR.

Both these tests use the normal audio path, thus also SCI_BASS, differential output mode, and EarSpeaker settings have an effect.

10.11.3 Pin Test

Pin test is activated with the 16-byte sequence 0x50 0xED 0x6E 0x54 0 0 0 0 0 0 0 0 0 0 0 0. This test is meant for chip production testing only.

10.11.4 SCI Test

Sci test is initialized with the 16-byte sequence 0x53 0x70 0xEE n 0 0 0 0 0 0 0 0 0 0 0 0, where n is the register number to test. The content of the given register is read and copied to SCI_HDAT0. If the register to be tested is HDAT0, the result is copied to SCI_HDAT1.

Example: if n is 0, contents of SCI register 0 (SCI_MODE) is copied to SCI_HDAT0.

10.11.5 Memory Test

Memory test mode is initialized with the 16-byte sequence 0x4D 0xEA 0x6D 0x54 0 0 0 0 0 0 0 0 0 0 0 0. After this sequence, wait for 1100000 clock cycles. The result can be read from the SCI register SCI_HDAT0, and 'one' bits are interpreted as follows:

Bit(s)	Mask	Meaning
15	0x8000	Test finished
14:10		Unused
9	0x0200	Mux test succeeded
8	0x0100	Good MAC RAM
7	0x0080	Good I RAM
6	0x0040	Good Y RAM
5	0x0020	Good X RAM
4	0x0010	Good I ROM 1
3	0x0008	Good I ROM 2
2	0x0004	Good Y ROM
1	0x0002	Good X ROM 1
0	0x0001	Good X ROM 2
	0x83ff	All ok

Memory tests overwrite the current contents of the RAM memories. A software or hardware reset continues normal operation.

11 VS8053b Registers

11.1 Who Needs to Read This Chapter

User software is required when a user wishes to add some own functionality like DSP effects to VS8053b.

However, most users of VS8053b don't need to worry about writing their own code, or about this chapter, including those who only download software plug-ins from VLSI Solution's Web site.

Note: Also see VS1063 Hardware Guide for more information, because the hardware is compatible with VS1053.

11.2 The Processor Core

VS_DSP is a 16/32-bit DSP processor core that also had extensive all-purpose processor features. VLSI Solution's free VSKIT Software Package contains all the tools and documentation needed to write, simulate and debug Assembly Language or Extended ANSI C programs for the VS_DSP processor core. VLSI Solution also offers a full Integrated Development Environment VSIDE for full debug capabilities.

11.3 VS8053b Hardware DAC Audio Paths

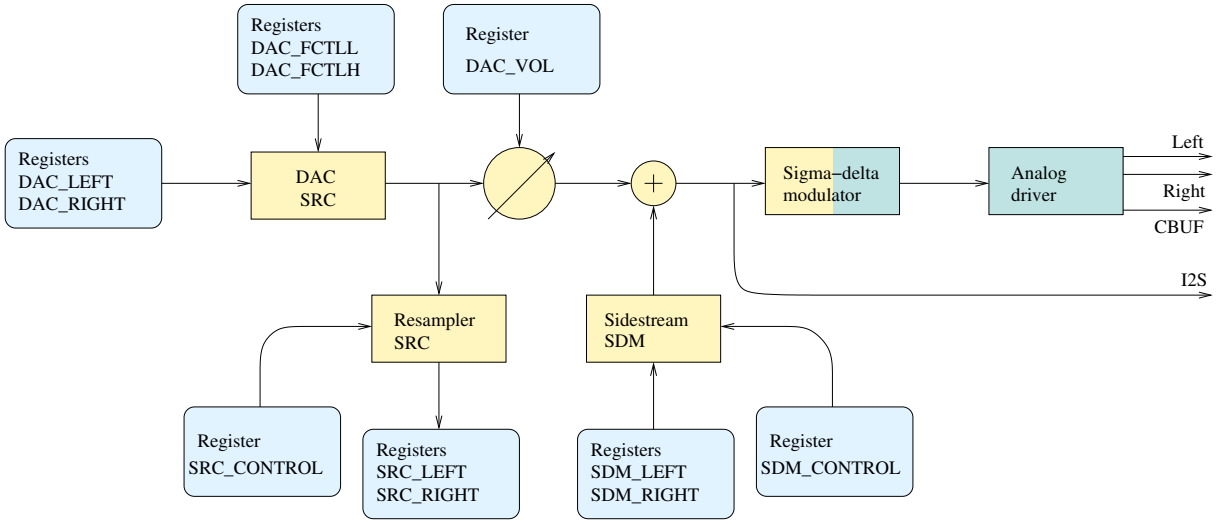


Figure 18: VS8053b ADC and DAC data paths with some data registers

Figure 18 presents the VS8053b Hardware DAC audio paths.

The main audio path starts from the DAC register (Chapter 11.8) to the high-fidelity, fully digital DAC SRC (Digital-to-Analog Converter SampleRate Converter), which low-pass filters and interpolates the data to the high samplerate of XTALI/2 (nominally 6.144 MHz). This 18-bit data is then fed to the volume control. It then passes through the sigma-delta modulator to the analog driver and analog Left and Right signals.

The user may resample and record the data with the Resampler SampleRate Converter (Chapter 11.16). Because there is no automatic low-pass filtering, it is the user's responsibility to avoid aliasing distortion.

The user may add a PCM sidestream with the Sidestream Sigma-Delta Modulator input (Chapter 11.17). As is the case with the Resampler SampleRate Converter, hardware doesn't offer low-pass filtering, so sufficient aliasing image rejection is the responsibility of the user.

11.4 VS8053b Hardware ADC Audio Paths

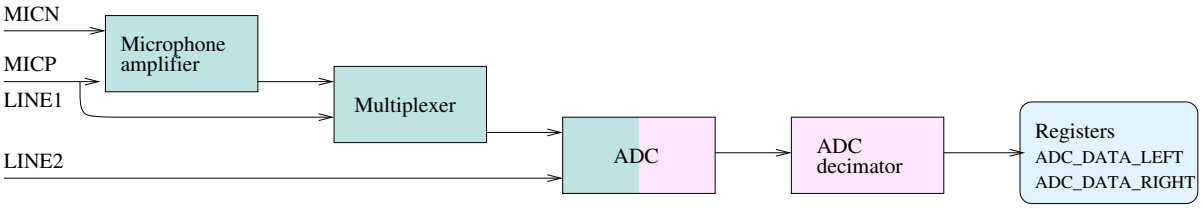


Figure 19: VS8053b ADC and DAC data paths with some data registers

Figure 18 presents the VS8053b Hardware ADC audio paths.

Analog audio may be fed up to two channels: one as a differential signal to MICN/MICP or as a one-sided signal to Line1, and the other as a one-sided signal to Line2.

If microphone input for the left channel has been selected, audio is fed through a microphone amplifier and that signal is selected by a multiplexer.

Audio is then downsampled to one of four allowed samplerates: XTALI/64, XTALI/128, XTALI/256 or XTALI/512. With the nominal 12.288 MHz crystal, these correspond to 192, 96, 48 or 24 kHz samplerates, respectively (Chapter 11.17).

If the “3 MHz” option bit SS_AD_CLOCK in register SCI_STATUS has been set to 1, then samplerates are divided by two, so the nominal samplerates become 96, 48, 24 and 12 kHz.

11.5 VS8053b Memory Map

X-memory		Y-memory		I-memory	
Address	**Description**	**Address**	**Description**	**Address**	**Description**
0x0000..0x17ff	System RAM	0x0000..0x17ff	System RAM	0x0000..0x004f	System RAM
0x1800..0x187f	User RAM	0x1800..0x187f	User RAM	0x0050..0x0fff	User RAM
0x1880..0x197f	Stack	0x1880..0x197f	Stack	0x1000..0x1fff	-
0x1980..0x3fff	System RAM	0x1980..0x3fff	System RAM	0x2000..0xffff	ROM 56k
0x4000..0xbfff	ROM 32k	0x4000..0xdfff	ROM 40k		and banked
0xc000..0xc0ff	Peripherals	0xe000..0xffff	System RAM	0xc000..0xffff	ROM4 16k
0xc100..0xffff	ROM 15.75k				

11.6 SCI Hardware Registers

SCI registers described in Chapter 9.6 can be found here between 0xC000..0xC00F. In addition to these registers, there is one in address 0xC010, called SCI_CHANGE.

SCI registers, prefix SCI_				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC010	r	0	CHANGE[5:0]	Last SCI access address

SCI_CHANGE bits		
Name	**Bits**	**Description**
SCI_CH_WRITE	4	1 if last access was a write cycle
SCI_CH_ADDR	3:0	SCI address of last access

SCI_CHANGE contains the last SCI register that has been accessed through the SCI bus, as well as whether the access was a read or write operation.

11.7 Serial Data Interface (SDI) Registers

Whenever two bytes have been written to the SDI bus, an interrupt is generated and the data can be read as a 16-bit big-endian value from the SDI registers. The user can control the DREQ pin as if it was a general-purpose output through its own register bit.

SDI registers, prefix SER_				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC011	r	0	DATA	Last received 2 bytes, big-endian
0xC012	w	0	DREQ[0]	DREQ pin control

11.8 DAC Registers

DAC registers, prefix DAC_				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC013	rw	0	FCTLL	DAC frequency control, 16 LSbs
0xC014	rw	0	FCTLH	DAC frequency control 4MSbs, PLL control
0xC015	rw	0	LEFT	DAC left channel PCM value
0xC016	rw	0	RIGHT	DAC right channel PCM value
0xC045	rw	0	VOL	DAC hardware volume

The internal 20-bit register DAC_FCTL is calculated from DAC_FCTLH and DAC_FCTLL registers as follows: DAC_FCTL = (DAC_FCTLH & 15) $\times$ 65536 + DAC_FCTLL. Highest supported value for DAC_FCTL is 0x80000.

If we define C = DAC_FCTL and X = XTALI in Hz, then the resulting samplerate f_s of the associated DAC SampleRate Converter is $f_s = C \times X \times 2^{-27}$.

Example:
If C = 0x80000 and X = 12.288 MHz then $f_s = 524288 \times (12.288 \times 10^6) \times 2^{-27} = 48000$ (Hz).

Note: FCTLH bits 13:4 are used for the PLL Controller. See Chapter 11.9 for details.

DAC_VOL bits		
Name	**Bits**	**Description**
LEFT_FINE	15:12	Left channel gain +0.0 dB. . .+5.5 dB (0 to 11)
LEFT_COARSE	11:8	Left channel attenuation in -6 dB steps
RIGHT_FINE	7:4	Right channel volume +0.0 dB. . .+5.5 dB (0 to 11)
RIGHT_COARSE	3:0	Right channel attenuation in -6 dB steps

Normally DAC_VOL is handled by the firmware. DAC_VOL depends on SCI_VOL and the bass and treble settings in SCI_BASS (and optionally SS_SWING bits in SCI_STATUS).

11.9 PLL Controller

The Phase-Locked Loop (PLL) controller is used to generate clock frequencies that are higher than the incoming (crystal-based) clock frequency. The PLL output is used by the CPU core and some peripherals.

Configurable features include:

- VCO Enable/Disable
- Select VCO or input clock to be output clock
- Route VCO frequency to output pin
- Select PLL clock multiplier

At the core of the PLL controller is the VCO, a high frequency oscillator, whose oscillation frequency is adjusted to be an integer multiple of some input frequency. As the name “Phase-Locked Loop” suggests, this is done by comparing the phase of the input frequency against the phase of a signal which is derived from the VCO output through frequency division.

If the system is stable, e.g. the comparison phase difference remains virtually zero, the PLL is said to be “in lock”. This means that the output frequency of the VCO is stable and reliable.

The PLL is preceded by a division-by-two unit. Thus, with a nominal XTALI = 12.288 MHz, the internal clock frequency CLKI can be adjusted with an accuracy of XTALI/2 = 6.144 MHz.

PLL control lies in DAC_FCTL bits 13:4. To see what bits 3:0 do, see Chapter 11.8.

FREQCTLH PLL bits, prefix FCH_		
Name	**Bits**	**Description**
PLL_LOCKED	13	0=lock failed since last test (read-only)
PLL_SET_LOCK	12	1:Sets FCH_PLL_LOCKED to 1 to start lock test
PLL_VCO_OUT_ENA	11	Route VCO to GPIO pin (VS1000:second cs pin)
PLL_FORCE_PLL	9	1:System clock is VCO / 0:System clock is inclk
PLL_DIV_INCLK	8	divide inclk by 2 (for 1.5, 2.5 or 3.5 x clk)
PLL_RATE	7:4	PLL rate control

The PLL locked status can be checked by generating a high-active pulse (writing first “1” , then “0”) to FCH_PLL_SET_LOCK and reading FCH_PLL_LOCKED. FCH_PLL_LOCKED is set to “1” along with the high level of FCH_PLL_SET_LOCK and to “0” whenever the PLL falls out of lock. So if the “1” remains in FCH_PLL_LOCKED, PLL is in sync.

The PLL controller’s operation is optimized for frequencies around 12. . . 13 MHz. If you use an 24. . . 26 MHz input clock, set the extra clock divider bit SM_CLK_RANGE in register SCI_MODE to 1 before activating the PLL.

It’s recommended to change the PLL rate in small steps and wait for the PLL to stabilize after each change. For diagnostic purposes, the PLL clock output (VCO) can be routed to an I/O pin so it can be scanned with an oscilloscope.

FCH_PLL_RATE (bits 7:4) control PLL multiplication rate. PLL multiplier is (FCH_PLL_RATE + 1). When FCH_PLL_RATE is 0, the VCO is powered down and output clock is forced to be input clock (same as if FCH_PLL_FORCE_PLL = 0).

11.10 GPIO

GPIO registers, prefix GPIO_				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC017	rw	0	DDR[9:0]	Direction
0xC018	r	0	IDATA[11:0]	Values read from the pins
0xC019	rw	0	ODATA[8:0]	Values set to the pins

GPIO_DIR is used to set the direction of the GPIO pins. 1 means output. GPIO_ODATA remembers its values even if a GPIO_DIR bit is set to input.

GPIO_IDATA is used to read the pin states. In VS8053 also the SDI and SCI input pins can be read through GPIO_IDATA: SCLK = GPIO_IDATA[8], XCS = GPIO_IDATA[9], SI = GPIO_IDATA[10], and XDCS = GPIO_IDATA[11].

In addition to data direction control for GPIO pins 0 to 7, there are two additional control bits in GPIO_DIR. GPIO_DIR[8] switches SO into software control, so that the value in GPIO_ODATA[8] is shown on SO whenever xCS is low. When GPIO_DIR[9] is set to '1' SCI and SDI are disabled. When using SCLK, xCS, SI and xDCS as general-purpose input, GPIO_DIR[9] prevents transitions in them from getting random data from SDI and/or SCI.

GPIO registers don't generate interrupts.

Note that in VS8053b the VSDSP registers can be read and written through the SCI_WRAMADDR and SCI_WRAM registers. You can thus use the GPIO pins quite conveniently.

11.11 Interrupt Control

Interrupt registers, prefix INT_				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC01A	rw	0	ENABLE[9:0]	Interrupt enable
0xC01B	w	0	GLOB_DIS[-]	Write to add to interrupt counter
0xC01C	w	0	GLOB_ENA[-]	Write to subtract from interrupt counter
0xC01D	rw	0	COUNTER[4:0]	Interrupt counter

INT_ENABLE controls the interrupts. The control bits are as follows:

INT_ENABLE bits		
Name	**Bits**	**Description**
INT_EN_SDM	9	Enable Sigma Delta Modulator interrupt
INT_EN_SRC	8	Enable SampleRate Converter interrupt
INT_EN_TIM1	7	Enable Timer 1 interrupt
INT_EN_TIM0	6	Enable Timer 0 interrupt
INT_EN_RX	5	Enable UART RX interrupt
INT_EN_TX	4	Enable UART TX interrupt
INT_EN_ADC	3	Enable AD modulator interrupt
INT_EN_SDI	2	Enable Data interrupt
INT_EN_SCI	1	Enable SCI interrupt
INT_EN_DAC	0	Enable DAC interrupt

Note: It may take up to 6 clock cycles before changing INT_ENABLE has any effect.

Writing any value to INT_GLOB_DIS adds one to the interrupt counter INT_COUNTER and effectively disables all interrupts. It may take up to 6 clock cycles before writing to this register has any effect.

Writing any value to INT_GLOB_ENA subtracts one from the interrupt counter INT_COUNTER, unless it already was 0, in which case nothing happens. If, after the operation INT_COUNTER becomes zero, interrupts selected with INT_ENABLE are restored. An interrupt routine should always write to this register as the last thing it does, because interrupts automatically add one to the interrupt counter, but subtracting it back to its initial value is the responsibility of the user. It may take up to 6 clock cycles before writing this register has any effect.

By reading INT_COUNTER the user may check if the interrupt counter is correct or not. If the register is not 0, interrupts are disabled.

11.12 UART (Universal Asynchronous Receiver/Transmitter)

The RS232 UART implements a serial interface using RS232 standard 8N1 (8 data bits, no parity, 1 stop bit).

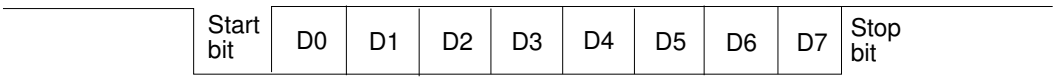


Figure 20: RS232 serial interface protocol

When the line is idling, it stays in logic high state. When a byte is transmitted, the transmission begins with a start bit (logic zero) and continues with data bits (LSB first) and ends up with a stop bit (logic high). 10 bits are sent for each 8-bit byte frame.

11.12.1 UART Registers

UART registers				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC028	r	0	UART_STATUS[4:0]	Status
0xC029	r/w	0	UART_DATA[7:0]	Data
0xC02A	r/w	0	UART_DATAH[15:8]	Data High
0xC02B	r/w	0	UART_DIV	Divider

11.12.2 Status UART_STATUS

A read from the status register returns the transmitter and receiver states.

UART_STATUS bits		
Name	**Bits**	**Description**
UART_ST_FRAMEERR	4	Framing error (stop bit was 0)
UART_ST_RXORUN	3	Receiver overrun
UART_ST_RXFULL	2	Receiver data register full
UART_ST_TXFULL	1	Transmitter data register full
UART_ST_TXRUNNING	0	Transmitter running

UART_ST_FRAMEERR is set if the stop bit of the received byte was 0.

UART_ST_RXORUN is set if a received byte overwrites unread data when it is transferred from the receiver shift register to the data register, otherwise it is cleared.

UART_ST_RXFULL is set if there is unread data in the data register.

UART_ST_TXFULL is set if a write to the data register is not allowed (data register full).

UART_ST_TXRUNNING is set if the transmitter shift register is in operation.

11.12.3 Data UART_DATA

A read from UART_DATA returns the received byte in bits 7:0, bits 15:8 are returned as '0'. If there is no more data to be read, the receiver data register full indicator is cleared.

A receive interrupt is generated when a byte is moved from the receiver shift register to the receiver data register.

A write to UART_DATA sets a byte for transmission. The data is taken from bits 7:0, other bits in the written value are ignored. If the transmitter is idle, the byte is immediately moved to the transmitter shift register, a transmit interrupt request is generated, and transmission is started. If the transmitter is busy, the UART_ST_TXFULL is set and the byte remains in the transmitter data register until the previous byte has been sent and transmission can proceed.

11.12.4 Data High UART_DATAH

The same as UART_DATA, except that bits 15:8 are used.

11.12.5 Divider UART_DIV

UART_DIV Bits		
Name	**Bits**	**Description**
UART_DIV_D1	15:8	Divider 1 (0..255)
UART_DIV_D2	7:0	Divider 2 (6..255)

The divider is set to 0x0000 in reset. The ROM boot code must initialize it correctly depending on the master clock frequency to get the correct bit speed. The second divider (D_2) must be from 6 to 255.

The communication speed $f = \frac{f_m}{(D_1+1)\times(D_2)}$, where f_m is the master clock frequency, and f is the TX/RX speed in bps.

Divider values for common communication speeds at 26 MHz master clock:

Example UART Speeds, $f_m = 49.152\,MHz$		
Comm. Speed [bps]	UART_DIV_D1	UART_DIV_D2
4800	255	40
9600	255	20
14400	233	15
19200	255	10
28800	243	7
38400	159	8
57600	121	7
115200	60	7

11.12.6 UART Interrupts and Operation

Transmitter operates as follows: After an 8-bit word is written to the transmit data register it is transmitted instantly if the transmitter is not busy transmitting the previous byte. When the transmission begins a TX_INTR interrupt is sent. Status bit [1] informs the transmitter data register empty (or full state) and bit [0] informs the transmitter (shift register) empty state. A new word must not be written to transmitter data register if it is not empty (bit [1] = '0'). The transmitter data register is empty as soon as it is shifted to transmitter and the transmission is begun. It is safe to write a new word to transmitter data register every time a transmit interrupt is generated.

Receiver operates as follows: It samples the RX signal line and if it detects a high to low transition, a start bit is found. After this it samples each 8 bit at the middle of the bit time (using a constant timer), and fills the receiver (shift register) LSB first. Finally the data in the receiver is moved to the reveive data register, the stop bit state is checked (logic high = ok, logic low = framing error) for status bit[4], the RX_INTR interrupt is sent, status bit[2] (receive data register full) is set, and status bit[2] old state is copied to bit[3] (receive data overrun). After that the receiver returns to idle state to wait for a new start bit. Status bit[2] is zeroed when the receiver data register is read.

RS232 communication speed is set using two clock dividers. The base clock is the processor master clock. Bits 15-8 in these registers are for first divider and bits 7-0 for second divider. RX sample frequency is the clock frequency that is input for the second divider.

11.13 Timers

There are two 32-bit timers that can be initialized and enabled independently of each other. If enabled, a timer initializes to its start value, written by a processor, and starts decrementing every clock cycle. When the value goes past zero, an interrupt is sent, and the timer initializes to the value in its start value register, and continues downcounting. A timer stays in that loop as long as it is enabled.

A timer has a 32-bit timer register for down counting and a 32-bit TIMER1_LH register for holding the timer start value written by the processor. Timers have also a 2-bit TIMER_ENA register. Each timer is enabled (1) or disabled (0) by a corresponding bit of the enable register.

11.13.1 Timer Registers

Timer registers, prefix TIMER_				
Reg	**Type**	**Reset**	**Abbrev**	**Description**
0xC030	r/w	0	CONFIG[7:0]	Timer configuration
0xC031	r/w	0	ENABLE[1:0]	Timer enable
0xC034	r/w	0	T0L	Timer0 startvalue - LSBs
0xC035	r/w	0	T0H	Timer0 startvalue - MSBs
0xC036	r/w	0	T0CNTL	Timer0 counter - LSBs
0xC037	r/w	0	T0CNTH	Timer0 counter - MSBs
0xC038	r/w	0	T1L	Timer1 startvalue - LSBs
0xC039	r/w	0	T1H	Timer1 startvalue - MSBs
0xC03A	r/w	0	T1CNTL	Timer1 counter - LSBs
0xC03B	r/w	0	T1CNTH	Timer1 counter - MSBs

11.13.2 Configuration TIMER_CONFIG

TIMER_CONFIG Bits		
Name	**Bits**	**Description**
TIMER_CF_CLKDIV	7:0	Master clock divider

TIMER_CF_CLKDIV is the master clock divider for all timer clocks. The generated internal clock frequency $f_i = \frac{f_m}{c+1}$, where f_m is the master clock frequency and c is TIMER_CF_CLKDIV. Example: With a 12 MHz master clock, TIMER_CF_DIV=3 divides the master clock by 4, and the output/sampling clock would thus be $f_i = \frac{12MHz}{3+1} = 3MHz$.

11.13.3 Configuration TIMER_ENABLE

TIMER_ENABLE Bits		
Name	**Bits**	**Description**
TIMER_EN_T1	1	Enable timer 1
TIMER_EN_T0	0	Enable timer 0

11.13.4 Timer X Startvalue TIMER_Tx[L/H]

The 32-bit start value TIMER_Tx[L/H] sets the initial counter value when the timer is reset. The timer interrupt frequency $f_t = \frac{f_i}{c+1}$ where f_i is the master clock obtained with the clock divider (see Chapter 11.13.2 and c is TIMER_Tx[L/H].

Example: With a 12 MHz master clock and with TIMER_CF_CLKDIV=3, the master clock $f_i = 3MHz$. If TIMER_TH=0, TIMER_TL=99, then the timer interrupt frequency $f_t = \frac{3MHz}{99+1} = 30kHz$.

11.13.5 Timer X Counter TIMER_TxCNT[L/H]

TIMER_TxCNT[L/H] contains the current counter values. By reading this register pair, the user may get knowledge of how long it takes before the next timer interrupt. Also, by writing to this register, a one-shot different length timer interrupt delay may be realized.

11.13.6 Timer Interrupts

Each timer has its own interrupt, which is asserted when the timer counter underflows.

11.14 I2S DAC Interface

The 16-bit I2S Interface makes it possible to attach an external DAC to the system.

Note: The samplerate of the audio file and the I2S rate are independent. All audio is automatically converted to 6.144 MHz for VS8053 DAC and to the configured I2S rate using a high-quality sample-rate converter.

Note: In VS8053b the I2S pins share different GPIO pins than in VS1033 to be able to use SPI boot and I2S in the same application.

I2S registers, prefix I2S_				
Reg	**Type**	**Reset**	**Abbrev**	**Description**
0xC040	r/w	0	CONFIG[3:0]	I2S configuration

I2S_CONFIG Bits		
Name	**Bits**	**Description**
I2S_CF_MCLK_ENA	3	Enables the MCLK output (12.288 MHz)
I2S_CF_ENA	2	Enables I2S, otherwise pins are GPIO
I2S_CF_SRATE	1:0	I2S rate, "10" = 192, "01" = 96, "00" = 48 kHz

I2S_CF_ENA enables the I2S interface. After reset I2S is disabled and the pins are used for GPIO inputs.

I2S_CF_MCLK_ENA enables the MCLK output. The frequency is either directly the input clock (nominal 12.288 MHz), or half the input clock when mode register bit SM_CLK_RANGE is set to 1 (24-26 MHz input clock).

I2S_CF_SRATE controls the output samplerate. When set to 48 kHz, SCLK is MCLK divided by 8, when 96 kHz SCLK is MCLK divided by 4, and when 192 kHz SCLK is MCLK divided by 2. I2S_CF_SRATE can only be changed when I2S_CF_ENA is 0.

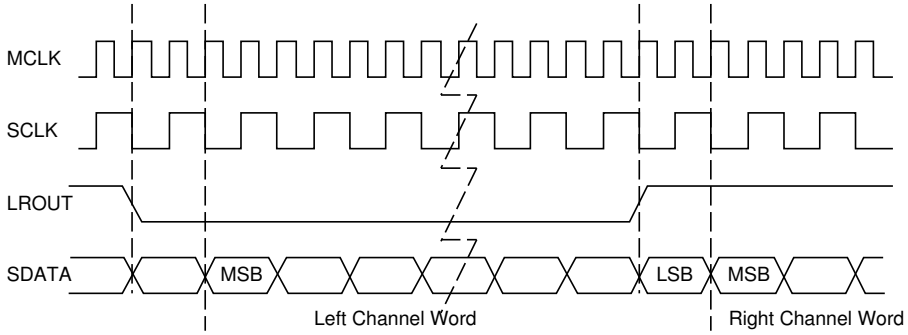


Figure 21: I2S interface, 192 kHz.

To enable I2S first write 0xc017 to SCI_WRAMADDR and 0x00f0 to SCI_WRAM, then write 0xc040 to SCI_WRAMADDR and 0x000c to SCI_WRAM.

11.15 Analog-to-Digital Converter (ADC)

ADC modulator registers control Analog-to-Digital conversions of VS8053b.

ADC Decimator registers, prefix ADC_				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC042	rw	0	CONTROL[4:0]	ADC control
0xC043	r	0	DATA_LEFT	ADC left channel data
0xC044	r	0	DATA_RIGHT	ADC right channel data

ADC_CONTROL controls the ADC and its associated decimator unit.

ADC_CONTROL Bits		
Name	**Bits**	**Description**
ADC_MODU2_PD	4	Right channel powerdown
ADC_MODU1_PD	3	Left channel powerdown
ADC_DECIM_FACTOR	2:1	ADC Decimator factor: - 3 = downsample to XTALI/512 (nominal 24 kHz) - 2 = downsample to XTALI/256 (nominal 48 kHz) - 1 = downsample to XTALI/128 (nominal 96 kHz) - 0 = downsample to XTALI/64 (nominal 192 kHz)
ADC_ENABLE	0	Set to activate ADC converter and decimator

Note: Setting bit SS_AD_CLOCK in register SCI_STATUS halves the operation speed of the A/D unit, and thus halve the resulting samplerate.

Each time a new (stereo) sample has been generated, an ADC interrupt is generated.

11.16 Resampler SampleRate Converter (SRC)

The resampler SRC makes it possible to catch audio from the DAC path.

Note: hardware makes no attempts at low-pass filtering data. If the SRC samplerate is lower than the DAC samplerate, aliasing may and will occur.

Resampler SRC registers, prefix SRC_				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC046	rw	0	CONTROL[12:0]	SRC control
0xC047	r	0	DATA_LEFT	SRC left channel data
0xC048	r	0	DATA_RIGHT	SRC right channel data

SRC_CONTROL Bits		
Name	**Bits**	**Description**
SRC_ENABLE	12	Set to enable SRC
SRC_DIV	11:0	Set samplerate to XTALI/2/(SRC_DIV+1)

Each time a new (stereo) sample has been generated, an SRC interrupt is generated.

11.17 Sidestream Sigma-Delta Modulator (SDM)

The Sidestream Sigma-Delta Modulator makes it possible to insert a digital side stream on top of existing audio.

Note: The SDM provides a direct, low-delay side channel to the Sigma-Delta DACs of VS10xx. It makes no attempts at low-pass filtering data. Thus there is practically no image rejection. If using low samplerates, this may cause audible aliasing distortion.

Sidestream SDM registers, prefix SDM_				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC049	rw	0	CONTROL[12:0]	SDM control
0xC04A	rw	0	DATA_LEFT	SDM left channel data
0xC04B	rw	0	DATA_RIGHT	SDM right channel data

SDM_CONTROL Bits		
Name	**Bits**	**Description**
SDM_ENABLE	12	Set to enable SDM
SDM_DIV	11:0	Set samplerate to XTALI/2/(SDM_DIV+1)

Each time a new (stereo) sample is needed, an SDM interrupt is generated.

12 Version Changes

This chapter describes the lastest and most important changes done to VS8053b

12.1 Differences Between VS1053b and VS8053b, 2010-04-30

VS8053b firmware uses laser fuse bits to disable MP3, WMA, AAC and Midi decoders. Otherwise the functionality and performance for VS1053b and VS8053b is exactly the same.

12.2 Changes Between VS1033c and VS1053a/b Firmware, 2007-03-08

Completely new or major changes:

- I2S pins are now in GPIO4-GPIO7 and do not overlap with SPI boot pins.
- No software reset required between files when used correctly.
- Ogg Vorbis decoding added. Non-fatal ogg or vorbis decode errors cause automatic resync. This allows easy rewind and fast forward. Decoding ends if the "last frame" flag is reached or SM_CANCEL is set.
- HE-AAC v2 Level 3 decoding added. It is possible to disable PS and SBR processing and control the upsampling modes through `parametric_x.control1`.
- Like the WMA decoder, the AAC decoder uses the clock adder (see SCI_CLOCKF) if it needs more clock to decode the file. HE-AAC features are dropped one by one, if the file can not be decoded correctly even with the highest allowed clock. Parametric stereo is the first feature to be dropped, then downsampled mode is used, and as the final resort Spectral Band Replication is disabled. Features are automatically restored for the next file.
- Completely new volume control with zero-cross detection prevents pops when volume is changed.
- Audio FIFO underrun detection (with slow fade to zero) instead of looping the audio buffer content.
- Average bitrate calculation (`byteRate`) for all codecs.
- All codecs support fast play mode with selectable speeds for the best-quality fast forward operation. Fast play also advances DECODE_TIME faster.
- WMA and Ogg Vorbis provide an absolute decode position in milliseconds.
- When SM_CANCEL is detected, the firmware also discards the stream buffer contents.
- Bit SCIST_DO_NOT_JUMP in SCI_STATUS is ’1’ when jumps in the file should not be done: during header processing.
- IMA ADPCM encode now supports stereo encoding and selectable samplerate.

Other changes or additions:

- Delayed volume and bass/treble control calculation reduces the time the corresponding SCI operations take. This delayed handling and the new volume control hardware prevents audio samples from being missed during volume change.
- SCI_DECODE_TIME only cleared at hardware and software reset to allow files to be played back-to-back or looped.

- Read and write to YRAM at 0xe000..0xffff added to SCI_WRAMADDR/SCI_WRAM.
- The `resync` parameter (parametric_x.resync) is set to 32767 after reset to allow inifinite resynchronization attempts (or until SM_CANCEL is set). Old operation can be restored by writing 0 to `resync` after reset.
- WMA,AAC: more robust resync.
- WMA,AAC: If resync is performed, broadcast mode is automatically activated. The broadcast mode disables file size checking, and decoding continues until SM_CANCEL is set or reset is performed.
- Treble control fixed (volume change could cause bad artefacts).
- MPEG Layer I mono fixed.
- MPEG Layer II half-rate decoding fixed (frame size was calculated wrong).
- MPEG Layer II accuracy problem fixed, invalid grouped values set to 0.
- WAV parser now skips unknown RIFF chunks.
- IMA ADPCM: Maximum blocksize is now 4096 bytes (4088 samples stereo, 8184 mono). Thus, now also plays 44100Hz stereo.
- Rt-midi: starts if in reset GPIO0='0', GPIO1='1', GPIO2&3 give earSpeaker setup.
- NewSinTest() and NewSinSweep() added (AIADDR = 0x4020/0x4022) AICTRL0 and AICTRL1 set sin frequency for left/right.
- Clears memory before SPI boot and not in InitHardware().

Known quirks, bugs, or features in VS1053b:

- Setting volume clears SS_REFERENCE_SEL and SS_AD_CLOCK bits. See Chapter 9.6.2.
- Software reset clears GPIO_DDR, also affects I2S pins.
- Ogg Vorbis occasionally overflows in windowing causing a small glitch to audio. Patch available (*VS1053b Patches w/ FLAC Decoder* plugin at *http://www.vlsi.fi/en/support/software/vs10xxpatches.html*).
- IMA ADPCM encoding requires short patch to start. Patch available in Chapter 10.8.1.
- If you want to start and end the sine test repeatedly, you need to send 12 zero bytes.
- There are also fixes for some other issues, we recommend you use the latest version of the *VS1053b Patches w/ FLAC Decoder* package from *http://www.vlsi.fi/en/support/software/vs10xxpatches.html*.

13 Latest Document Version Changes

This chapter describes the latest and most important changes to this document.

Version 1.33, 2025-04-04

- Old sine test needs 12 zero bytes sent to SDI, if you want to stop and start a new one repeatedly. See Chapter 10.11.1.

Version 1.32, 2024-02-01

- Updated Chapter 3, *Definitions*.
- Removed some references to unique ID, which is not generally available.

Version 1.31, 2017-11-17

- Split table in Chapter 4.3, *Analog Characteristics*, into two.
- Improved imafix in Chapter 10.8.1.

Version 1.30, 2016-12-21

- Updated numbers in Chapter 4, *Characteristics & Specifications*, according to re-qualification results.
- Added mention of 8N1 format to Chapter 11.12, *UART (Universal Asynchronous Receiver/Transmitter)*.

Version 1.22, 2014-12-19

- Added mention of RIFF 8-bit and 16-bit data signedness to Chapter 10.6, *Feeding PCM Data*.
- Updated telephone number in Chapter 14, *Contact Information*.

Version 1.21, 2014-08-13

- Clarified how Ogg Vorbis set Replay Gain parameters in Chapter 10.10.2, *Ogg Vorbis*.
- Explained that Bass Enhancer handles the bass part of the audio signal in mono in Chapter 9.6.3, *SCI_BASS*.
- Added GPIO_DDR[9:8] and GPIO_ODATA[8] explanation to Chapter 11.10, *GPIO*.
- Moved value from tV(min) to tV(max) in Chapter 7.4.4, *SPI Timing Diagram*.
- Chapter 10.8, *PCM / ADPCM Recording*, has been updated to better explain how to record using the VS1053b Patches package.
- Clarified byte endianness when reading PCM or ADPCM audio data in Chapter 10.8.3, *Adding a PCM RIFF Header*, and Chapter 10.8.4, *Adding an IMA ADPCM RIFF Header*.

14 Contact Information

VLSI Solution Oy
Entrance G, 2nd floor
Hermiankatu 8
FI-33720 Tampere
FINLAND

URL: http://www.vlsi.fi/
Phone: +358-50-462-3200
Commercial e-mail: sales@vlsi.fi

For technical support or suggestions regarding this document, please participate at
http://www.vsdsp-forum.com/
For confidential technical discussions, contact
support@vlsi.fi